VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

APLIKACE PRO REPORTOVÁNÍ DOPRAVNÍCH PŘESTUPKŮ NA IOS

NÁZEV PRÁCE(ANGLICKY)

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **TOMÁŠ BUDÁČ**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. PETR BOBÁK**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2020/2021

Zadání bakalářské práce

Student: **Budáč Tomáš**
Program: Informační technologie
Název: **Aplikace pro reportování dopravních přestupků na iOS**
iOS Application for Traffic Violation Reporting
Kategorie: Uživatelská rozhraní

Zadání:

1. Seznamte se s architekturou iOS, jazykem Swift a frameworky SwiftUI a UIKit.
2. Podrobněji analyzujte možnosti akvizice obrazu na zařízeních se systémem iOS.
3. Prozkoumejte podobné existující aplikace a proveďte jejich analýzu.
4. Nastudujte designové principy návrhu uživatelského rozhraní pro platformu iOS a iterativním způsobem navrhněte uživatelské rozhraní aplikace.
5. Navrženou aplikaci implementujte.
6. Otestujte funkcionalitu a použitelnost výsledné aplikace.
7. Zhodnoťte dosažené výsledky, vytvořte plakát a krátké prezentační video. Dále navrhněte možné pokračovaní.

Literatura:

- Human Interface Guidelines: https://developer.apple.com/design/human-interface-guidelines/
- SwiftUI: https://developer.apple.com/xcode/swiftui/
- AVFoundation framework: https://developer.apple.com/documentation/avfoundation
- dále dle pokynů vedoucího

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Bobák Petr, Ing.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 30. října 2020

Abstrakt

Táto práca sa zaoberá návrhom a implementáciou aplikácie pre operačný systém iOS a serveru REST API. Aplikácia je implementovaná v programovacom jazyku Swift a jeho rozhraní SwiftUI. Skladá sa z teoretickej a praktickej časti. V teoretickej časti sa zameriava na popis použitých technológií a aktuálny stav už existujúcich riešení. Praktická časť obsahuje návrh, implementáciu a testovanie aplikácie vytvorené na základe poznatkov získaných z teoretickej časti. Výsledkom je užívateľsky prívetivá mobilná aplikácia pre nahlasovanie dopravných priestupkov spojených s nesprávnym parkovaním v mestách s webovým rozhraním pre správu priestupkov.

Abstract

This bachelor thesis is dedicated to the proposal and implementation of the application for both the iOS operating system and the server REST API. The application was implemented in the programming language Swift and its interface SwiftUI. This thesis is structured into a theoretical section, which focuses on the description of current technologies and the current state of already existing solutions. The practical section incorporates the proposal, implementation and app trial. All of that was created based on theoretical knowledge. The result is a user-friendly mobile application for reporting traffic offenses associated with inappropriate parking in the cities with a web-based traffic management interface.

Kľúčové slová

iOS, mobilná aplikácia, Apple, SwiftUI, Swift, MapKit, MVVM, dopravný priestupok, ARKit, REST API, Vapor

Keywords

iOS, mobile application, Apple, SwiftUI, Swift, MapKit, MVVM, traffic offense, ARKit, REST API, Vapor

Citácia

BUDÁČ, Tomáš. *Aplikace pro reportování dopravních přestupků na iOS*. Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Petr Bobák

Aplikace pro reportování dopravních přestupků na iOS

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením Ing. Petra Bobáka. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Tomáš Budáč
10. mája 2021

Poďakovanie

Ďakujem pánovi inžinierovi Petrovi Bobákovi za odborné usmernenia, cenné rady a vedenie práce.

Obsah

Kapitola 1

Úvod

V súčasnej veľmi uponáhľanej dobe sa chytrý mobilný telefón stal každodennou súčasťou života a osvojil si pomenovanie smartfón. Neslúži už len ako prostriedok komunikácie, ale dokáže nahrádzať niektoré funkcie osobného počítača alebo hernej konzoly. Dôležitou zložkou smartfónu je operačný systém. Na trhu sú dvaja najväčší súperi iOS a Android. Operačný systém iOS vznikol v spoločnosti Apple, ktorá neumožňuje jeho použitie inde ako na svojich zariadeniach. Za operačným systémom Android stojí spoločnosť Google a ide o otvorený softvér dostupný pre každého.

Cieľom tejto práce je vytvoriť mobilnú aplikáciu na operačný systém iOS pre používateľov na nahlasovanie dopravných priestupkov spojených s parkovaním. Taktiež používateľovi priniesť zjednodušenie procesu nahlasovania priestupkov. Aplikácia musí byť používateľsky prívetivá a schopná odosielať priestupky ďalej na spracovanie. Rozšírením tejto práce je taktiež serverová aplikácia, ktorá ponúka webové rozhranie pre správu priestupkov pridelenou zodpovednou osobou.

Druhá kapitola 2 práce sa zaoberá vývojom aplikácií pre operačný systém iOS a serverových aplikácií, rozšírenou realitou a popisom použitých frameworkov alebo nástrojov. Tretia kapitola 3 analyzuje existujúce riešenia. V štvrtej kapitole 4 sa nachádza návrh aplikácie. Rozoberá sa tu funkcionalita jednotlivých prvkov aplikácie s ich použitím, návrh používateľského rozhrania v jednotlivých iteráciách spoločne s návrhom databázy. Piata kapitola 5 sa zaoberá implementáciou mobilnej a serverov aplikácie. Šiesta kapitola 6 popisuje spôsoby testovania aplikácií a vyhodnocovanie výsledkov testovania. V závere je zhrnuté hodnotenie aplikácií, ich ďalšie použitie a rozšírenia.

Kapitola 2

Súčasný stav technológií

Táto kapitola stručne popisuje nástroje pre návrh a implementáciu aplikácií. Následne analyzuje a popisuje už existujúce riešenia aplikácií, ktoré slúžia k nahláseniu podnetov. Bližšie sa zameriava na možnosť nahlásenia dopravných priestupkov spojených s nesprávnym parkovaním.

2.1 Vývoj aplikácií pre operačný systém iOS

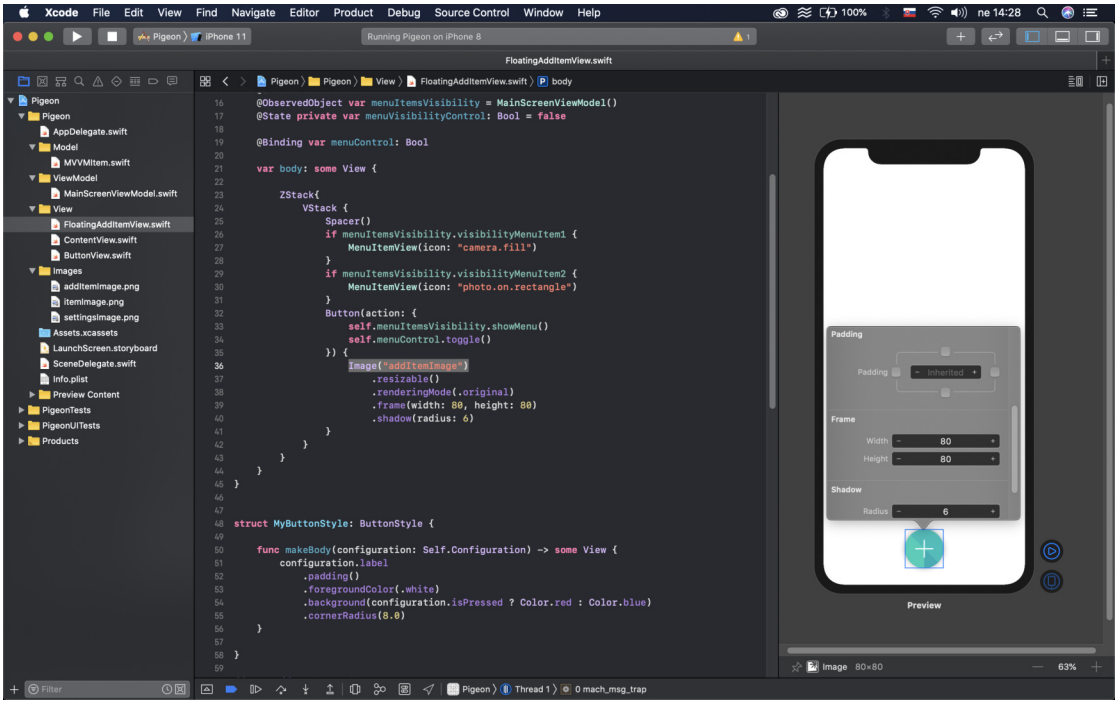
Pre možnosť vývoja pre platformu iOS je potrebné spĺňať určité kritéria. Najdôležitejším je vlastníctvo zariadenia s operačným systémom macOS, kde vývoj bude prebiehať. Následne je potrebné vlastniť účet v spoločnosti Apple. Ten povoľuje prístup na sťahovanie aplikácií z App Store. Pre publikovanie aplikácií v App Store je nevyhnutné k účtu prikúpiť rozšírený vývojársky balík[1] za 99$.

Na vývoj aplikácie sa využíva integrované vývojarské prostredie Xcode, ktoré je dostupné zadarmo v App Store. To obsahuje najnovšie vývojárske nástroje, editor kódu, editor grafického používateľského rozhrania, ladiace nástroje a mnoho ďalších. Obsahuje aj simulátor zariadení na testovanie aplikácií bez ich fyzickej potreby. Po splnení týchto kritérií je vývojár schopný vytvárať aplikácie pre všetky typy zariadení spoločnosti Apple.

2.2 Programovací jazyk Swift

Najpoužívanejším programovacím jazykom pre vývoj iOS aplikácií je v súčasnosti Swift[12]. Predstavuje výkonný a intuitívny programovací jazyk pre systémy macOS, iOS, tvOS, iPadOS a watchOS. Vznikol výsledkom prieskumov o najnovších programovacích jazykoch v spojení s rokmi skúseností vo vývoji pre platformu Apple. Parametre funkcií a metód sú vyjadrené prehľadnou syntaxou, ktorá zjednodušuje čitateľnosť kódu aplikácie. Automatické odvodenie typov tvoria kód čistejším a menej náchylným na chyby. Pre lepšiu podporu medzinárodných jazykov a emodži sú reťazce v Unicode a na optimalizáciu výkonu v širokej škále prípadov použitia používajú kódovanie založené na UTF-8. Pamäť sa riadi pomocou deterministického počítania referencií. Táto technika automaticky uvoľní pamäť používanú inštanciami triedy, ktoré nie sú potrebné. Tým minimalizuje využitie pamäte.[8]

[1] https://developer.apple.com/support/compare-memberships/



Obr. 2.1: Ukážka využitia frameworku SwiftUI vo vývojovom prostredí Xcode

2.3 Rozhranie SwiftUI

SwiftUI bol predstavený na konferencii WWDC[2] v roku 2019 ako nový framework[3] prevývoj aplikácií pre zariadenia Apple. Pre mobilné zariadenia je potrebná minimálna verzia iOS 13. Považuje sa za nástupcu v súčasnosti najpoužívanejšieho frameworku UIKit.

SwiftUI slúži na tvorbu používateľského rozhrania. Umožňuje vývoj používateľských rozhraní *deklaratívnym* spôsobom. Týmto spôsobom je možné meniť naraz všetky stavy hodnôt medzi viacerými používateľskými rozloženiami. Obsahuje vlastné zobrazenia, ovládacie prvky a štruktúry rozloženia. Podporuje *drag and drop*. Jednotlivé prvky je možné meniť aj na ukážke rozhrania. Zmeny sú synchronizované s kódom v priľahlom editore (Obrázok 2.1 vpravo). Taktiež všetky zmeny kódu sú okamžite viditeľné na ukážke rozhrania. Xcode okamžite prekompiluje všetky zmeny a vloží ich do bežiacej verzie aplikácie, ktorá je vždy viditeľná a upraviteľná. Podporuje možnosť integrácie objektov z UIKit, AppKit a WatchKit frameworkov.[5]

Deklaratívne programovanie je programovanie paradigmou, kde ide o vytváranie štruktúr a prvkov počítačového programu, ktoré vyjadrujú logiku výpočtu bez popísania riadiaceho toku. Zjednodušene je to programovanie pomocou definícií, čo sa má urobiť a nie ako sa to má urobiť. Opakom je imperatívne programovanie, ktoré využíva príkazy meniace stav programu napríklad po stlačení tlačidla je vykonaná funkcia alebo sada inštrukcií. Zameriava sa na opis fungovania programu, kde jednotlivé úlohy sú popísané pomocou algoritmov. V deklaratívnom rozhraní sa jednotlivým komponentom nenastavuje kedy sa majú zobrazovať alebo skrývať, ale nastavujú sa pravidlá pre ich riadenie. To však zapríčiňuje menšiu kontrolu.[17]

[2] Apple Worldwide Developer Conference – Každoročná konferencia pre vývojárov Apple
[3] https://whatis.techtarget.com/definition/framework

Hlavné obmedzenia SwiftUI:

- **Pokrytie API** – Nemá rovnako široké pokrytie API ako UIKit. Určité úlohy v SwiftUI sú ťažké až nemožné na vyriešenie, napríklad nie je možné do upozornenia pridať textové pole. Oproti tomu UIKit často ponúka pre dané úlohy triviálne riešenia.
- **Adaptácia** – SwiftUI nepodporuje spätnú kompatibilitu s nižšími verziami ako iOS 13. Takmer každá aplikácia doteraz používa UIKit. Aplikácie by mali mať podporu aj na staršie verzie systému. Príkladom je iOS 12 alebo iOS 11. Z toho vyplýva, že prechodom na SwiftUI by aplikácie neboli spustiteľné na zariadeniach s verziou operačného systému staršou ako iOS 13.
- **Podpora** – UIKit existuje viac ako desať rokov, čo znamená, že takmer každý problém je už vyriešený. Taktiež existuje veľa knižníc, ktoré poskytujú rôzne rozšírenia a prispôsobenia. SwiftUI má k dispozícii podstatne menej riešení, pretože je o dosť novší. Je zvyčajné hľadať riešenia, ktoré ešte nikto neskúšal. To môže spôsobiť väčšiu časovú náročnosť projektov.[13]

2.4 Zásady používateľského rozhrania v iOS

Návrh unikátneho vzhľadu aplikácie, ktorý dostane aplikáciu na vrchol rebríčkov, podlieha náročným očakávaniam na kvalitu a funkcionalitu.

Hlavné zásady vychádzajúce z príručky [7]:

- **Prehľadnosť** – Naprieč celým systémom je text čitateľný v každej veľkosti, ikony sú presné a prehľadné, ozdoby sú jemné a vhodné, vzhľad je zameraný na funkčnosť. Prázdne miesta, farby, písma, grafiky a prvky rozhrania jemne zvýrazňujú dôležitý obsah a približujú interakciu.
- **Rozloženie** – Plynulé pohyby a pekné rozhranie pomáhajú používateľovi porozumieť práci s obsahom. Obsah vypĺňa celú obrazovku. Rozmazanie a priesvitnosť naznačujú prvky na interakciu. Na zachovanie jemného a vzdušného rozhrania sa používa minimálne množstvo rámov, prechodov a tieňov.
- **Hĺbka** – Realistický pohyb a odlišné vizuálne vrstvy sprostredkúvajú hierarchiu a prístup k funkčnosti k ďalšiemu obsahu bez straty kontextu.

Vzhľadové zásady na maximalizovanie dosahu vlastnej aplikácie[7]:

- **Estetická integrita** – Predstavuje do akej miery sa vzhľad a správanie aplikácie integrujú do jej funkcionality.
- **Konzistencia** – Implementuje známe štandardy a paradigmy pomocou systémových prvkov rozhrania.
- **Priama manipulácia** – Priamo spája používateľa s obsahom na obrazovke a uľahčuje mu porozumieť obsahu. Používateľ vidí okamžité výsledky svojich akcií.
- **Spätná väzba** – Poskytuje spätnú odozvu na každú akciu používateľa. Zvýraznenie, animácie a zvuk pomáhajú objasniť výsledky akcií.
- **Metafory** – Používateľ sa učí rýchlejšie, ak virtuálne objekty a akcie aplikácie sú mu predom známe zo skutočného alebo digitálneho sveta.

- **Kontrola používateľa** – V celom systéme iOS má kontrolu používateľ, nie aplikácie. Aplikácia môže upozorniť alebo navrhnúť postup na nebezpečné akcie s následkami. Môže priniesť pocit, že používateľ má kontrolu udržovaním interaktívnych a známych prvkov, potvrdením deštruktívnych akcií a zrušením prebiehajúcich akcií. Cieľom je vytvoriť rovnováhu medzi povoleniami používateľa a zabránením nechcených výsledkov.

2.5 Metódy návrhu používateľského rozhrania

Používateľské rozhranie tvorí dôležitú rolu v softvéri. Rozhranie určuje aké množstvo informácií sú pre používateľa viditeľné. Každý malý prvok návrhu používateľského rozhrania môže ovplyvniť softvér pozitívne alebo negatívne. Preto sa používajú rôzne metodiky na návrh.

Organizačné schémy

Organizačné schémy súvisia s tým, ako sa bude kategorizovať obsah a vytvárať vzťahy medzi každou časťou systému. Väčšinu obsahu je možné kategorizovať viacerými spôsobmi. Schémy je možné rozdeliť na presné a subjektívne. V závislosti od obsahu je možné, že rozhranie kombinuje rôzne schémy a zaobchádza s nimi inými spôsobmi.

Presné organizačné schémy rozdeľujú informácie do navzájom sa vylučujúcich častí. Pre navrhovateľa schémy je ľahké jej vytvorenie a kategorizovanie, ale od používateľa sa vyžaduje znalosť, ako hľadať v tomto modeli.

Subjektívne organizačné schémy kategorizujú informácie konkrétne definované organizáciou alebo odborom. Ich navrhnutie je zložitejšie, ale oproti presným organizačným schémam sú často užitočnejšie. Pri návrhu sa zvažuje používateľov mentálny model a zmysluplné zoskupenie obsahu. Výsledkom je uľahčenie používateľovi porozumieť a nájsť spojenia medzi rôznymi časťami obsahu.

Organizačné štruktúry

Organizačné schémy sú spôsob definovania vzťahov medzi časťami obsahu. Úspešné štruktúry umožňujú používateľovi napovedať, kde hľadanú informáciu nájde. Je dôležité zohľadňovať očakávania používateľov, vytvárať konzistentné metódy organizovania a zobrazovať informácie tak, aby si používateľ rozšíril svoje vedomosti zo známych stránok na neznáme. Tri hlavné organizačné štruktúry sú hierarchická, sekvenčná a maticová.

Hierarchické štruktúry sú označované ako stromové štruktúry, kde prístup medzi informáciami je zhora nadol alebo vzťah rodiča a dieťaťa. Používatelia začínajú hľadanie od všeobecnejších kategórií (rodič) a potom nižšie hľadajú podrobnejšie informácie (dieťa). S touto štruktúrou sa používateľ stretáva denne napríklad v práci, kde sa používa vodcovská štruktúra.

Sekvenčné štruktúry vyžadujú, aby používatelia postupovali krok po kroku a sledovali konkrétnu cesty cez obsah. Príkladom je nakupovanie položiek cez internet, kde je používateľ musí vyplniť všetky údaje a potvrdiť objednávku.

Maticová štruktúra dovoľuje používateľom určiť si vlastnú cestu, pretože obsah je prepojený rôznymi spôsobmi. Tento typ štruktúry plne využíva princípy hypertextu alebo HTML. Jeden používateľ si napríklad môže zvoliť navigáciu v obsahu založenú na dátume, zatiaľ čo iný navigáciu založenú na téme.

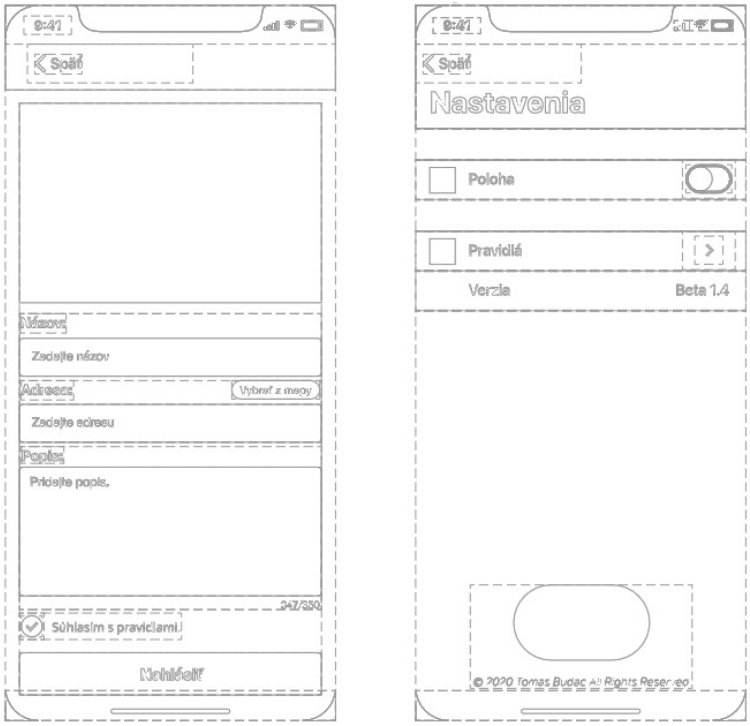
Prvky používateľského rozhrania

Pri navrhovaní používateľské rozhrania treba byť pri výbere prvkov dôsledný a predvídateľný. Jednotlivé prvky musia používateľovi byť známe alebo vhodne popisovať význam. Voľba vhodných prvkov pomôže k dokončeniu úlohy jednoduchšie a účinnejšie.

Prvky používateľského rozhrania sú napríklad ovládače vstupov, navigačné alebo informačné komponenty a kontajnery.

Wireframy

Wireframe je dvojrozmerná ilustrácia používateľského rozhrania (Obrázok 2.2), ktorá sa zameriava na umiestnenie v priestore a určenie priorít obsahu, dostupné funkcionality a zamýšľané fungovanie. Na základe týchto dôvodov wireframy neobsahujú žiadny štýl, farbu ani grafiku. Taktiež pomáhajú zobrazovať vzťahy medzi rôznymi wireframami.[21]



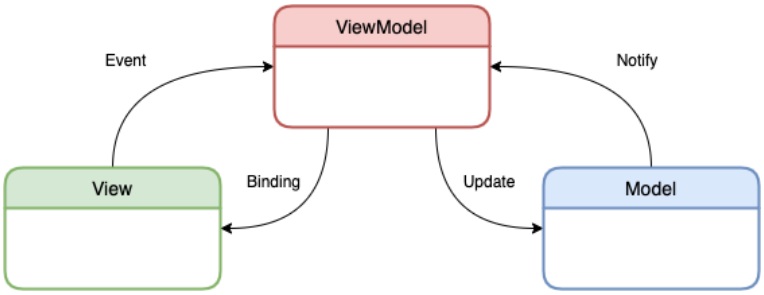
Obr. 2.2: Wireframy obrazoviek iOS aplikácie

2.6 Návrhový vzor Model–View–ViewModel (MVVM)

Návrhový vzor Model–View–ViewModel označovaný ako MVVM ponúka riešenie, ako oddeliť logiku aplikácie od používateľského rozhrania. MVVM oddeľuje dáta, stav aplikácie a používateľské rozhranie. Výsledný kód sa stáva prehľadnejším a prípadné zmeny sú implementačné nenáročné. Používa sa vo SwiftUI.

Skladá sa z vrstiev:

- **Model** – Popisuje dáta, s ktorými aplikácia pracuje. Zvyčajne sa jedná o štruktúry a jednoduché triedy.



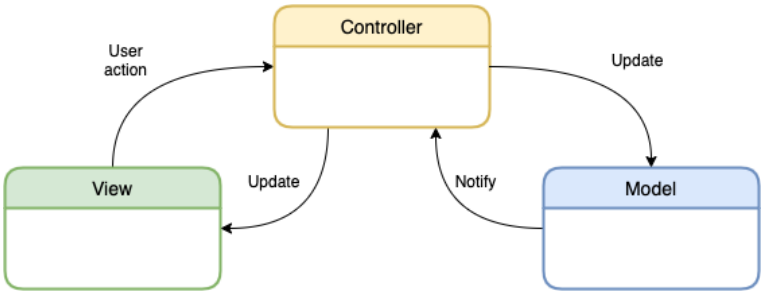
Obr. 2.3: Diagram návrhového vzoru Model–View–ViewModel

- **View** – Zobrazuje vizuálne a ovládacie prvky na obrazovke. Označuje sa ako používateľské rozhranie.
- **ViewModel** – Spája Model a View. Pretvára modelové informácie na zobrazované hodnoty. Tvorí hlavnú logiku aplikácie.

Úlohou je vytvoriť triedu, ktorá drží stav aplikácie. Nazýva sa ViewModel. Zodpovedá sa používateľskému rozhraniu, ktoré podľa nej vykresľuje ovládacie prvky. Ak používateľ zmení dáta v používateľskom rozhraní, tak dáta sa automaticky odovzdajú do ViewModelu. Toto spojenie funguje pomocou *binding*. ViewModel poskytuje všetky potrebné dáta pre používateľské rozhranie View. Svoje dáta zdieľa v dátových štruktúrach, ktoré pri ich zmene vyvolajú udalosti. Zabezpečuje to používateľskému rozhraniu automaticky zobraziť nové dáta, ktoré boli zmenené.[23]

2.7 Porovnanie návrhových vzorov MVC a MVVM

MVC je architektonický vzor (Obrázok 2.4), ktorý rozdeľuje aplikácie na tri hlavné logické komponenty Model, View a Controller. Plné znenie MVC je Model View Controller. Každý komponent architektúry je zostavený tak, aby zvládol špecifické vývojové aspekty aplikácie. Oddeľuje od seba obchodnú logiku a prezentačnú vrstvu. Taktiež ako MVVM (Sekcia 2.6) sa používa hlavne na grafické používateľské rozhrania.



Obr. 2.4: Diagram návrhového vzoru Model–View–Controller

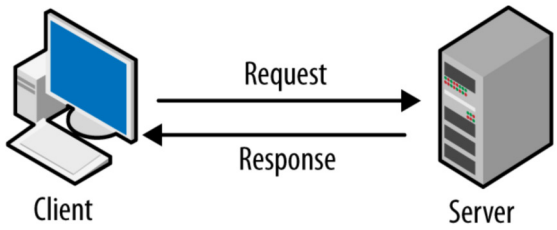
Rozdiely medzi MVVM a MVC:

- MVC je architektonický vzor, ktorý rozdeľuje aplikácie na tri hlavné logické komponenty. Na druhej strane MVVM umožňuje oddelenie vývoja grafického rozhrania pomocou značkovacieho jazyka alebo kódu rozhrania.

- V MVC je vstupným bodom aplikácie Controller, zatiaľ čo v MVVM je vstupom View.
- MVC neobsahuje samostatnú komponentu pre spravovanie rozhrania alebo prezenčnej logiky. Tým nedovoľuje jednoduché testovanie jednotlivých komponentov nezávisle na sebe. Naopak, implementácia jednotkových testov pre komponenty MVVM je jednoduchá a kód je riadený udalosťami.
- Vzťahy jeden k mnohým v MVC sú medzi Controllerom a View. V MVVM medzi View a ViewModel.

2.8 Architektúra klient-server a API

Termín klient-server sa používa na opis výpočtového modelu pre vývoj počítačového systému (Obrázok 2.5). Model je založený na rozdelení funkcií medzi dva nezávislé autonómne procesy server a klient.



Obr. 2.5: Model klient-server prevzaté z príručky [2]

Klient je proces, ktorý vyžaduje špecifické služby od servera. Server poskytuje požadované služby pre klienta a neustále čaká na požiadavky. Svoje vstupy a výstupy má presne špecifikované. Procesy medzi klientom a serverom môžu prebiehať na rovnakom zariadení alebo na rozličných zariadeniach prepojených sieťou. Ďalej spravuje používateľské rozhranie a logiku aplikácie, prijíma a kontroluje syntax vstupov používateľa, vytvára a odosiela databázové požiadavky na server.

Server môže poskytovať svoje služby viacerým klientom nezávislých na sebe. Služby môže vykonávať paralelne alebo sekvenčne podľa jeho implementácie. Klient môže požadovať služby z niekoľkých serverov v sieti bez ohľadu na ich umiestnenie a vlastnosti zariadenia. Sieť spája klienta so serverom ako médium, cez ktoré medzi sebou komunikujú. Prijíma a spracúva databázové požiadavky od klienta a odpovedá mu, zaisťuje integritu a kontroluje autorizáciu.

Architektúra klient-server obsahuje rôzne typy vrstiev.

Jednovrstvová architektúra

Architektúra prvej úrovne obsahuje konfiguračné nastavenia klienta a servera, používateľské rozhranie, dátovú logiku a obchodnú logiku v rovnakom systéme. Tieto typy služieb sú spoľahlivé, ale je ich veľmi ťažké udržiavať, pretože obsahujú údaje v rôznych formátoch, ktoré sú určené pre fungovanie systému.

Napríklad prezentačná, obchodná, dátová vrstva používajú jeden softvérový balík. Všetky dáta sú uložené na lokálnom zariadení.

Dvojvrstvová architektúra

Dvojvrstvová architektúra poskytuje najlepšie prostredie, kde používateľské rozhranie je uložené na strane klienta a všetky dáta sa ukladajú na serveri. Obchodná a databázová logika môžu byť vytvorené u klienta, ale aj na serveri. Keď sa na klientskej strane riadi databázová a obchodná logika, tak sa jedná o architektúru tenkého servera a tučného klienta. Ak sú však obchodná a databázová logika riadené serverom, potom je to známa ako architektúra tučného servera tenkého klienta.

V tejto architektúre sú klient a server priamo prepojené, teda medzi nimi nie je žiadny sprostredkovateľ. Výsledkom je výstup s najvyššou rýchlosťou a ignorujú sa nedorozumenia medzi ostatnými klientmi.

Trojvrstvová architektúra

V trojvrstvovej architektúre je potrebný middlevér, pretože spracúva klientsku požiadavku na strednej vrstve. Tu vytvorí požiadavku server, ktorú server následne získa, spracuje a odpovie na ňu. Takže prvá odpoveď servera je prijatá strednou vrstvou, potom je získaná klientom. Celá dátová a obchodná logika sú uložené v middlevéri. Použitie middlevéru zlepšuje flexibilitu a zabezpečuje lepší výkon.

Trojvrstvová architektúra je rozdelená do prezentačnej, aplikačnej a databázovej vrstvy. Klient spracúva prezentačnú vrstvu, middlevér kontroluje aplikačnú vrstvu a server sa stará o databázovú vrstvu.

Viacvrstvová architektúra

Viacvrstvová architektúra má zmenšenú formu trojvrstvovej architektúry. V tejto architektúre sú prezentácie, spracovanie aplikácií a funkcie správy údajov navzájom izolované.[11]

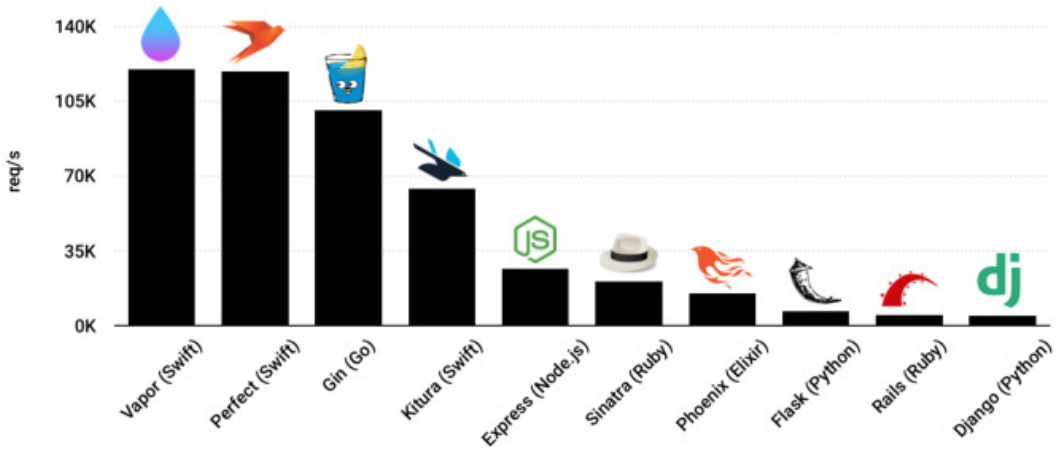
2.8.1 Výhody a nevýhody architektúry klient-server

Hlavnou výhodou je výkon a znížené zaťaženie. Spracovanie sa distribuuje medzi klienta a server. Strana klienta nemusí spracovávať procesy u seba na zariadení. Naopak server nemusí poskytovať užívateľské rozhranie. Nezávislosť na pracovnej stanici a interoperabilita systému používateľa neobmedzuje iba na jeden typ systému alebo platformy. Ďalej zachováva integritu údajov, čo umožňuje systému poskytovať množstvo služieb, ktoré chránia dáta, šifrujú úložisko a zálohujú ho v reálnom čase. V prípade zlyhania systému sú zmeny vykonané v databáze zachované.

Medzi hlavné nevýhody patria náklady na správu a pomocný personál pri údržbe databázového servera. Ďalej sú náklady na školenia pre personál, hardvér, softvérové licencie a oprava porúch pri komplexnejších systémoch.[19]

2.8.2 REST API

REST API je aplikáčné programové rozhranie, ktoré podlieha obmedzeniam architektúry REST[4]. Niekedy pomenované ako RESTful API. Umožňuje interakciu s webovými službami REST. API označuje súbor definícií a protokolov na vytváranie a integráciu aplikačného softvéru. Taktiež sa označuje ako zmluva medzi poskytovateľom a používateľom informácií alebo služieb zabezpečujúcich požadujúci obsah spotrebiteľa (hovor) a obsah požadovaný výrobcom (odpoveď). Zjednodušene API je sprostredkovateľ medzi používateľmi alebo klientami a zdrojmi alebo webovými službami, ktoré chcú získať. Je to spôsob, ktorým orga-



Obr. 2.6: Porovnanie frameworkov v počte požiadavkov, prevzaté z [18]

nizácia môže zdieľať zdroje a informácie. Zachováva bezpečnosť, kontrolu a autentifikáciu, ktorou určuje úroveň prístupnosti. Architektúra sa skladá z klientov, serverov a prostriedkov s požiadavkami spravovanými pomocou HTTP.

Ak je požiadavka klienta vytvorená pomocou rozhrania RESTful API, prenesie sa reprezentácia dotazu na žiadateľa alebo na koncový bod. Reprezentácie informácií sa doručujú cez protokol HTTP, napríklad pomocou formátov JSON, HTML alebo čistého textu. Najbežnejším formátom je JSON kvôli jeho dobrej čitateľnosť pre ľudí a stroje.

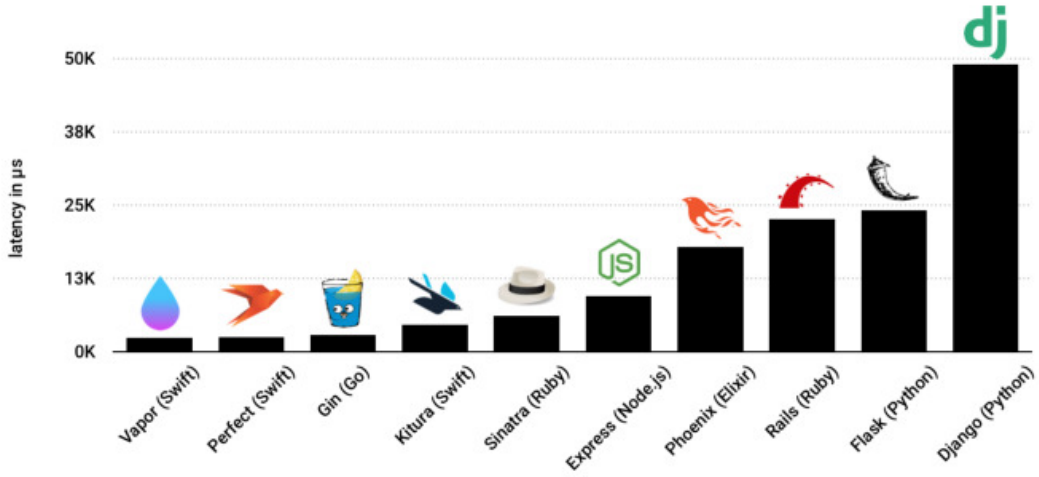
HTTP metódy obsahujú hlavičky a parametre. Ďalej obsahujú dôležité informácie o identifikátore pre žiadosť metadát, autorizácii, jednotnom identifikátore zdroja (URI), ukladaní do pamäte cache, dát typu cookies a podobne. Existujú aj hlavičky požiadaviek a odpovedí, ktoré obsahujú vlastné stavové kódy a informácie o HTTP pripojení.[16]

2.8.3 Serverová časť vo Vapor

Vapor je webový framework pre Swift (popísaný v sekcii 2.2), ktorý umožňuje vytvárať RESTful API a webové aplikácie. Funguje na systémoch macOS a Ubuntu. Obsahuje balíčky pre autentifikáciu, vývoj užívateľského rozhrania a správu databáz, napríklad SQL, Postgres a MongoDB. Hlavnou výhodou je programovací jazyk Swift, ktorý dovoľuje využitie svojich funkcií a možnosť kódovať alebo dekódovať štruktúry z alebo do HTTP správ. V základe sa používa kódovanie JSON s podporou multipart[5]. Rozhranie API je konfigurovateľné, čo umožňuje pridávať, upravovať alebo nahrádzať stratégie kódovania pre niektoré typy HTTP obsahu.

V porovnaní serverových frameworkov v počte požiadaviek za sekundu, kde viac je lepšie, dosahuje najlepšie výsledky oproti konkurencii (Obrázok 2.6). To isté dokazuje graf 2.7 pre priemernú latenciu pre každú požiadavku za jednotkový čas, čím nižšie číslo tým lepšie.[18]

[4]skratka pre prenos reprezentačného stavu
[5]viacdielna správa obsahujúca dáta

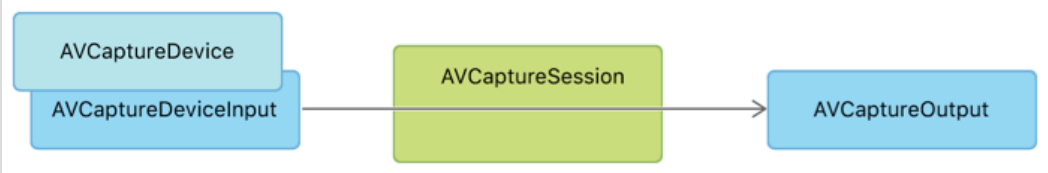


Obr. 2.7: Porovnanie frameworkov pre priemernú latenciu, prevzaté z [18]

2.9 Akvizícia obrazu

Akvizícia obrazu je proces získania obrazu z kamery zariadenia a obraz je prevedený do entity, s ktorou je možno ďalej pracovať. Získavanie obrazu pozostáva z energie odrážajúcej sa od objektu záujmu, optického systému, ktorý zameriava energiu a snímača meracieho množstvo energie. Energia v tomto kontexte predstavuje svetlo alebo elektromagnetické vlny.[20]

Pre zariadenia od spoločnosti Apple sa o snímanie videí, fotografií a zvuku stará AVFoundation Capture subsystém. Poskytuje vývojárom prístup k práci s fotoaparátom a jeho konfigurácii. Ďalej umožňuje vytvárať vlastné používateľské rozhranie fotoaparátu, kde sú možnosti pre priamu kontrolu nad zaostrením, expozíciou a stabilizáciou. Taktiež poskytuje živý prístup k obrazovým alebo zvukovým údajom získaných priamo zo snímacieho zariadenia.



Obr. 2.8: Diagram architektúry snímania prevzaté z príručku [4]

Hlavné časti architektúry snímania sú relácie, vstupy a výstupy (Obrázok 2.8). Snímacie relácie spájajú jeden alebo viacero vstupov do jedného alebo viacerých výstupov. Vstupy sú zdroje médií vrátane snímacích zariadení, ako sú vstavané fotoaparáty a mikrofóny zariadení s operačným systémom iOS. Výstupy získavajú médiá zo vstupov, kde produkujú užitočné dáta, ako sú napríklad video súbory zapísané na disk alebo surové pixely v medzipamäti pre živé spracovanie.

Systémová aplikácia kamery zariadení s operačným systémom iOS umožňuje zachytávať fotografie a videá z predných a zadných kamier. Taktiež podporuje statické snímanie hĺbkových údajov, efekty portrétu a živé fotografie, ak to zariadenie podporuje.[4]

Proces snímania jedného obrázka sa nazýva expozícia. Vzťahuje sa aj na množstvo svetla na jednotku plochy. Toto množstvo svetla musí byť v určitom rozmedzí. Ak nie je zachytený dostatok svetla, obraz bude podexponovaný, teda výsledný obraz bude tmavý. Ak zachytíme príliš veľa svetla, obraz bude preexponovaný, teda obraz bude príliš svetlý. Snímač obrazu je vtedy presýtený a už nedokáže rozlišovať medzi rôznymi množstvami svetla, čím vytvára pre všetky oblasti rovnakú expozíciu. Pri fotografovaní je potrebné nastaviť množstvo svetla dostatočne vysoké, ale nie až príliš vysoké.

Parametre ovplyvňujúce množstvo svetla expozície sú rýchlosť uzávierky, hodnota ISO a clona. Vo fotografii zmena ktoréhokoľvek z týchto parametrov, kde sa množstvo svetla zdvojnásobí alebo o polovicu zmenší sa nazýva jedna zastávka. Pre každý parameter jedna zastávka znamená zmenu jeho hodnoty. Ak sa nastaví rýchlosť uzávierky o jednu zastávku, tak je potrebné kompenzovať nastavenie ISO alebo clonu presne o jednu zastávku, aby bolo dosiahnuté rovnaké množstvo svetla. Zložitou časťou je, že všetky tri parametre ovplyvňujú aj ďalšie aspekty expozície. Existuje množstvo kombinácií týchto troch parametrov, ktoré vedú k rovnakému množstvu svetla.

Pri snímaní obrazu snímač zachytáva svetlo za určitý čas. Toto trvanie sa nazýva rýchlosť uzávierky, pretože popisuje, ako rýchlo sa uzávierka otvára a zatvára. Hodnota ISO sa nazýva aj rýchlosť filmu. Je to miera citlivosti obrazového snímača na svetlo, teda ako bude rušivá expozícia. Clona od objektívu fotoaparátu je mierkou veľkosti otvoru, cez ktorý svetlo dopadá na obrazový snímač. Clona je určená clonovým číslom, ktoré odpovedá pomeru ohniskovej vzdialenosti k efektívnemu priemeru clony.

Fotoaparát dokáže ostro vykresliť iba položky, ktoré sú v určitej vzdialenosti od fotoaparátu. Položky v rozsahu sú zaostrené. Položky príliš blízko alebo ďaleko nie sú zaostrené. Väčšina fotoaparátov vrátane iOS zariadení má automatické zaostrovanie. Fotoaparát odhadne, ktorá časť obrázka by mala byť ostrá a podľa toho upraví svoje zaostrenie. Vstavaná aplikácia fotoaparátu pre iOS zariadenia umožňuje používateľom stlačiť na miesto, čím sa zaostrí na túto časť obrázka. Niektoré aplikácie umožňujú používateľovi aj manuálne zaostrenie.[10]

2.10 Rozšírená realita

Rozšírená realita, často označovaná ako AR[6], je umiestňovanie počítačom generovaného obrazu do skutočného sveta pomocou kamery a obrazovky smartfónu. Často sa porovnáva s virtuálnou realitou VR[7]. Pričom virtuálna realita vytvára plne simulované prostredie, v ktorom sa užívateľ môže pohybovať. Základnou vlastnosťou AR je schopnosť vytvárať a sledovať závislosti medzi skutočným priestorom a virtuálnym priestorom, kde používateľ môže modelovať vizuálny obsah.

Vhodným príkladom rozšírenej reality je hra Pokemon Go[8], ktorá obsahuje režim AR. Umožňuje používateľovi loviť, fotografovať a chytať Pokémona cez kameru smartfónu. Platformy sociálnych sietí používajú filtre rozšírenej reality aby podporovali zdieľanie fotografií a videí. Ďalej je to aplikácia iOS Measure[9] umožňujúca merať vzdialenosť medzi objektami v reálnom priestore. Je vytvorená pomocou frameworku ARKit popísaného v sekcii 2.10. Niektoré obchody ponúkajú možnosť prehliadať si položky do domácnosti pomocou roz-

[6] augmented reality
[7] virtual reality

šírenej reality. Zákazníkovi to dáva lepšiu predstavu o umiestnení položky v interiéri ešte pred jej samotnou kúpou.

V súčasnosti sa AR primárne zameriava na smartfóny a tablety. Do budúcnosti sa odborníci prikláňajú k vyrábaniu AR okuliarov alebo náhlavných súprav.[15]

ARKit

ARKit je framework od spoločnosti Apple. Taktiež softvérová vývojárska súprava používaná vývojármi na vytváranie hier a nástrojov pre rozšírenú realitu na iOS. Prvýkrát bol predstavený na konferencii WWDC 2017, kde priblížili vývojárom prácu s ARKitom.[6]

Pre vytvorenie rozšírenej reality, ďalej označovanú ako AR, ARKit využíva svetové a kamerové súradnicové systému. Riadi sa pravostrannou konvenciou, kde os y smeruje nahor, os z smeruje k používateľovi a os x smeruje doprava od používateľa. Konfigurácia relácií môže zmeniť pôvod a orientáciu súradnicového systému vzhľadom na skutočný svet.

Kotva je objekt ktorý definuje pozíciu a orientáciu položky v skutočnom svete. AR konfigurácia obsahuje viaceré kotvy, ktoré sa riadia pravostrannou konvenciou. Napríklad trieda `ARFaceAnchor` definuje systém na lokalizáciu tvárových prvkov.

Technika pre vytváranie korešpondencie medzi skutočným a virtuálnym priestorom pre ARKit sa nazýva vizuálna inerciálna odometria. Tento proces kombinuje informácie z hardvéru iOS zariadenia na snímanie pohybu s analýzou počítačového videnia scény viditeľnej z kamery zariadenia. Rozpoznáva špecifické prvky scény, sleduje rozdiely v pozíciach týchto prvkov vo videozáberoch a porovnáva tieto informácie s údajmi o snímaní pohybu.

Na hľadanie povrchov v reálnom svete zodpovedajúcich bodu v obraze kamery sa používa metóda vysielania lúčov. Konfigurácia dovoľuje povoliť detekciu povrchov, kde ARKit detekuje ploché povrchy v obraze z kamery. Návratovými hodnotami je ich pozícia a veľkosť. Výsledkom tejto detekcie je možnosť umiestňovania alebo interakcie s virtuálnym obsahom scény.

Detekovanie povrchov je výpočtovo náročné pre hardvér zariadenia. Analýza obrazu vyžaduje jasný obraz a dobré svetelné podmienky. Ak je scéna príliš tmavá, tak vzdialenosť detekcie je len niekoľko centimetrov až sa nemusí vôbec podariť. Nadmerný pohyb alebo silné trasenie kamerou vedie k rozmazanému obrazu a tým znižuje dosah AR scény.

Prvotné načítanie scény môže udávať nepresnú polohu a dosah. Každou ďalšou iteráciou snímania sa spresňujú informácie o scéne. Objekty do scény je možné vkladať až po vytvorení scény. Ak scéna nebude obsahovať dostatok informácií o objektoch, vložené objekty do scény nebudú presne uložené. Väčšinou platí, čím dlhší čas na načítanie scény, tým presnejšie umiestnenie objektu.[9]

[8] https://pokemongolive.com/en/
[9] https://apps.apple.com/us/app/measure/id1383426740

Kapitola 3

Analýza existujúcich riešení

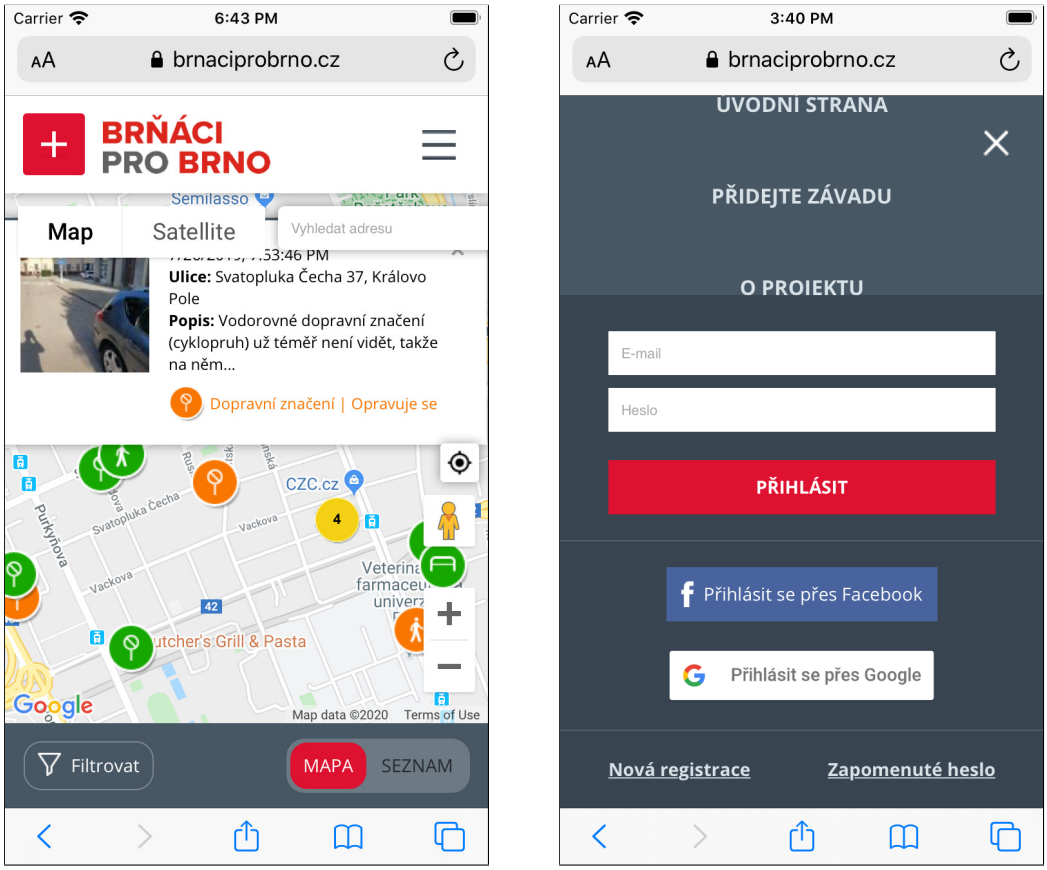
Táto kapitola popisuje aplikácie, ktoré slúžia k zaznamenávaniu podnetov a dopravných priestupkov spojených s parkovaním. Zameriava sa na tri aplikácie. Tie boli vybrané na základe internetového prieskumu a vlastných skúseností. V meste Brno sa najčastejšie používa webová aplikácia Brňáci pro Brno[1]. V Prahe Změňte.to[2], ktorá má vytvorenú mobilnú a webovú aplikáciu. Medzi vyhľadávanými aplikáciami v Českej republike nebola nájdená žiadna aplikácia priamo zameraná na nahlásenie dopravných priestupkov spojených s parkovaním. Jediným možným spôsobom je kontaktovať príslušný zodpovedný orgán v danej lokalite. Rozšírením prieskumu o Spojené Kráľovstvo došlo k nájdeniu aplikácie na nahlásenie nevhodne stojacich vozidiel. Táto aplikácia sa nazýva i-Ticket[3].

3.1 Brňáci pro Brno

Aplikácia spoločnosti Brněnské komunikace a.s. umožňuje občanom mesta Brno upozorniť na poruchy na miestnych komunikáciách alebo na cestách druhej a tretej triedy, ktoré sú v jej správe. Taktiež na poruchy na prvkoch verejného osvetlenia, ktoré sú pod správou spoločnosti Technické sítě Brno, a.s.. Nahlásené poruchy preverí príslušný technik a v prípade oprávnenosti požiadavky zaistí opravu.

Po otvorení úvodnej obrazovky aplikácie sa zobrazí mapa s miestami porúch (Obrázok 3.1 vľavo). Tie po otvorení zobrazujú podrobnosti o poruche. Ďalej umožňuje zobraziť zoznam porúch, ktorý môže byť filtrovaný. Vo vrchnej časti obrazovky sa nachádza navigačný panel s rýchlym presmerovaním na nahlásenie poruchy a menu. Menu (Obrázok 3.1 vpravo) ponúka prechod na úvodnú stranu, pridanie poruchy, informácie o projekte a možnosť prihlásenia. Možnosť prechodu na úvodnú stranu je nepotrebná, pretože menu je možné ukončiť stlačením ikony zavretia. Prvá položka zoznamu menu je nesprávne zarovnaná na vrch. Kvôli tomu používateľ nevidí správne obsah zoznamu. Pohyb po mape nie je plynulý a načítanie porúch pri pomalom internete trvá dlhší čas. Pri vytváraní poruchy je používateľ schopný nahrať len jednu fotografiu, ktorú môže orezať. Ďalším krokom používateľ označí miesto poruchy na mape. Následne musí vybrať typ poruchy a môže ju popísať. Pred odoslaním k spracovaniu má na výber, či chce byť informovaný o zmene stavu poruchy prostredníctvom e-mailu.

[1] https://www.brnaciprobrno.cz/akce/mapa
[2] https://apps.apple.com/us/app/zmente-to/id1210177921
[3] https://www.uk-carparkmanagement.co.uk/report-a-vehicle



Obr. 3.1: Webová aplikácia Brňáci pro Brno – zľava: Úvodná obrazovka, menu

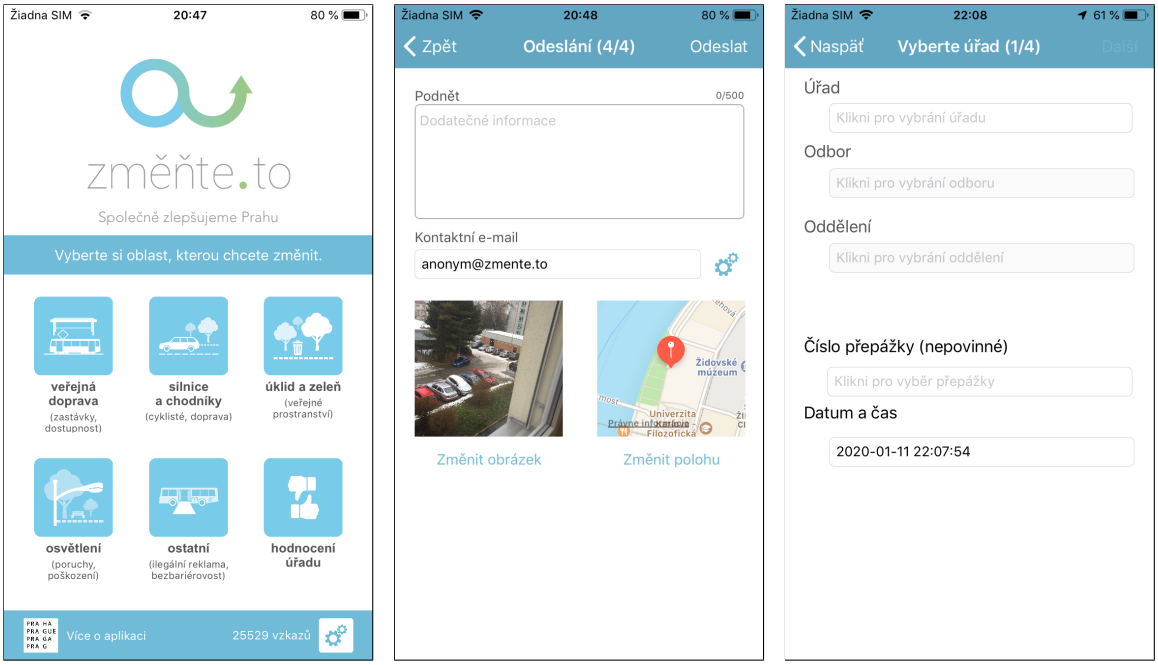
Od ostatných aplikácií sa líši zobrazením všetkých porúch na mape. Vytvoriť hlásenie o poruche je možné v troch krokoch. V niektorých prípadoch aplikácia nie je plynulá. Neobsahuje možnosť nahlásiť nevhodne stojace vozidlá.

3.2 Změňte.to

Tento pražský projekt umožňuje verejnosti posielať návrhy, podnety, pochvaly a sťažnosti o miestnej doprave zamestnancom lokálneho magistrátu alebo jej organizáciám. Podnet je možné nahlásiť pomocou webovej stránky alebo mobilnej aplikácie.

Po zapnutí mobilnej aplikácie sa zobrazí hlavná obrazovka, ktorú vidieť na obrázku 3.2 vľavo. Ponúka na výber päť rôznych typov podnetov s ich krátkym popisom a jednu možnosť na hodnotenie úradu. Ďalej obsahuje nastavenia a informácie o aplikácii. Využíva jednoduchý a prehľadný vzhľad. Ikony vystihujú ich podstatu. Pri rýchlej interakcii s prvkami ikon ukazujúcimi na typ podnetu sa obrazovky navzájom prekrývajú. Tým vytvoria viaceré vrstvy, ktoré používateľ musí zavrieť manuálne. Pri správnom návrhu aplikácie by sa takému správaniu anticipovalo. Pri vytváraní podnetu sa ako prvá zaznamenáva fotografia, ktorú je možné vybrať z galérie alebo ju zaznamenať. Nasleduje kontrola, či naozaj fotografia zreteľne zachytáva podnet. Ďalej nasleduje výber polohy z mapy pomocou GPS alebo jej manuálnym zadaním. Aplikácia umožní používateľovi prejsť na nasledujúci krok bez zadania polohy alebo vytvorenia fotografie. Posledný krok obsahuje dodatočný popis

podnetu, voliteľný kontaktný e-mail, možnosť zmeny obrázku a polohy. Používateľovi je umožnené nahrať len jednu fotografiu. Pri pokuse nahlásiť podnet bez vyplnenia kľúčových informácií je podnet nahlásený alebo aplikácia nečakane skončí. Takéto správanie nesmie nastať. Používateľ má byť upozornený na chýbajúce údaje a nasmerovaný na ich doplnenie. Nastavenia ponúkajú zadanie e-mailu pre zasielanie informácií o nahlásenom podnete. Tým dostáva používateľ spätnú väzbu o jeho podnete.



Obr. 3.2: Mobilná iOS aplikácia změňte.to – zľava: Hlavná obrazovka, posledný krok vytvorenia podnetu, nastavenia

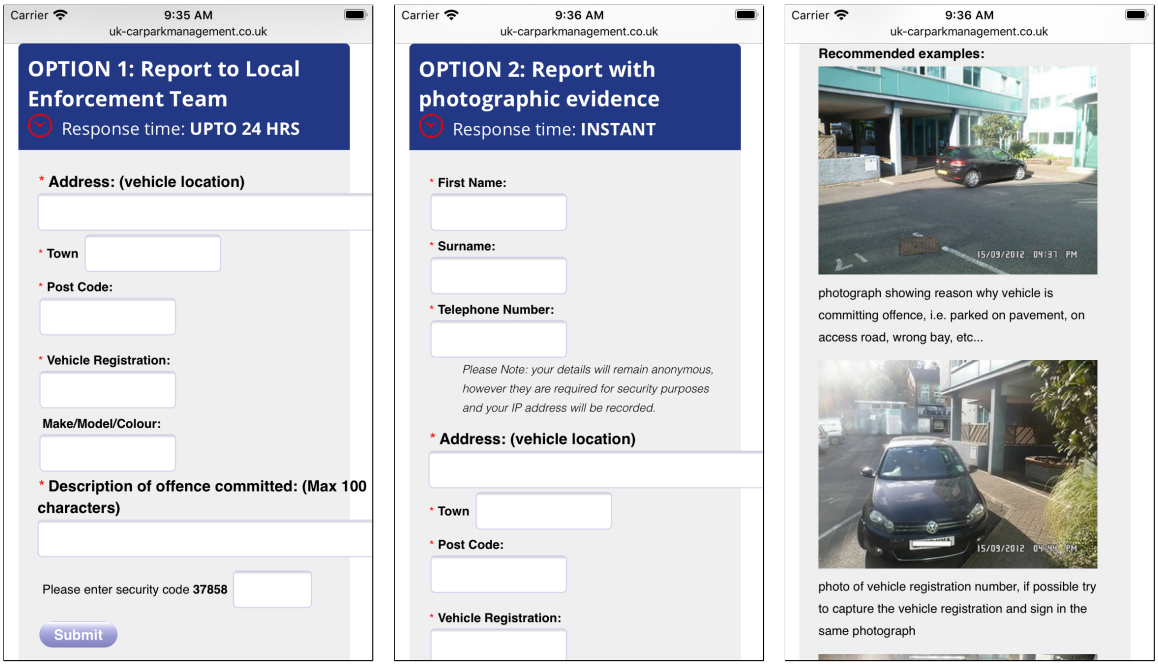
Ide o zaujímavý projekt, ktorého cieľom je zlepšiť verejné blaho života v Prahe. Mobilná aplikácia obsahuje nedostatky, ktoré bežný používateľ nemôže akceptovať. Výsledkom je negatívna spätná väzba a zlé hodnotenie v obchode App Store. Aplikácia nespĺňa zásady používateľského rozhrania.

3.3 i-Ticket

Ide o aplikáciu fungujúcu v Spojenom Kráľovstve cielenú pre menšie podniky, ktoré majú problém s nedostatkom parkovacích miest. Tie sú určené pre zákazníkov daného podniku. Ľudia zapĺňajú voľné parkovacie miesta podniku bez toho, aby boli zákazníkmi daného podniku. Napriek tomu, podniky si nemôžu dovoliť súkromné služby alebo detekčné systémy.

Pre nahlásenie priestupku za parkovanie je potrebná registrácia podniku do systému. Registrácia vyžaduje údaje o podniku, následne je možné nahlásiť priestupok. Potrebná je dokumentácia fotografiou, číslo ŠPZ, adresa a prípadný popis. Priestupok bude vybavený do piatich dní od nahlásenia. Za každý nahlásený priestupok, ktorý bude zaplatený zodpovednou osobou, používateľ dostane £10.

Webová aplikácia ponúka dve možnosti nahlásenia priestupkov. Prvou možnosťou je nahlásenie lokálnemu výkonnému tímu, ktorú vidieť na obrázku 3.3 vľavo. Reakčný čas odozvy obyčajne býva do 24 hodín. Potrebné sú údaje o presnej adrese, poznávacie číslo



Obr. 3.3: Webová časť Report a vehicle aplikácie UK Car Park Management – zľava: Možnosť nahlásenia lokálnej správe, možnosť nahlásenia spolu s fotografiou, príklad fotografie s popisom

vozidla, popis priestupku a voliteľné dodatočné informácie o vozidle. Na ošetrenie proti spamu je potrebné prepísať vygenerovaný bezpečnostný kód. Pre používateľa je toto riešenie najjednoduchšie. Pre určeného kontrolóra prichádza problém v podobe hľadania vozidla bez dokumentácie fotografiou, čo neprináša priamy dôkaz o spáchaní priestupku. Taktiež sa jedná o zdĺhavý proces hľadania konkrétneho vozidla. V inom prípade tam vozidlo už nemusí byť odstavené a kontrolór plytvá svojím časom. Táto metóda kontroly je neefektívna z hľadiska času a financií.

Druhou možnosťou je nahlásenie s priloženou fotografiou, ktorú je vidieť na obrázku 3.3 v strede a vpravo. Obsahuje rovnaké potrebné údaje ako prvá možnosť a naviac dodatočné informácie o vozidle. Tieto údaje sú rozšírené o celé meno a telefónne číslo podávateľa a čas zaznamenania priestupku. Prílohu tvorí jedna až tri fotografie daného vozidla. Toto prináša zdĺhavejšie vypĺňanie pre používateľa, ale presnejší popis pre kontrolóra. Vytvorené hlásenie sa okamžite zaeviduje do systému správy a kontrolór alebo príslušný orgán obdrží nahlásenie. Podľa zákonov Spojeného Kráľovstva je toto nahlásenie v súlade so zákonom a môže byť použité ako dôkazový materiál k spáchanému priestupku. Z toho vyplýva, že zodpovednej osobe za priestupok hrozí finančná pokuta.

3.4 Zhrnutie nedostatkov a návrhy pre vylepšenie

Hlavným problémom prvých dvoch aplikácií spomínaných vyššie je obmedzenie lokality na jedno konkrétne mesto. Taktiež pri rýchlej interakcii s prvkami zobrazenia aplikácia nebola plynulá. Obe aplikácie nepodporujú možnosť nahlásenia dopravných priestupkov. Posledná aplikácia pre bežného používateľa neponúka mobilnú verziu a je viazaná zákonmi jednej krajiny.

Pri vytváraní hlásení neboli niektoré povinné polia vyplnené, ale hlásenia boli napriek tomu vytvorené. V prípade viacerých nahlásení žiadna aplikácia nezaznamenáva všetky hlásenia bez registrácie. Ďalej je vhodné previesť nového používateľa aplikáciou pri prvom spustení a zoznámiť ho s pravidlami. Prípadne mu spríjemniť používanie aplikácie rôznymi plynulými animáciami.

Kapitola 4

Návrh aplikácie

V tejto kapitole je popísaný návrh aplikácie. Popisuje funkcionality aplikácie, cieľovú skupinu používateľov, prípady použitia a návrh používateľského rozhrania. Táto kapitola taktiež obsahuje charakteristiku funkcionality aplikácie.

4.1 Popis funkcionality aplikácie

Aplikácia má za úlohu byť používateľsky veľmi jednoduchou, aby nevznikol žiadny problém pri jej prvotnom použití. Musí dbať na základné princípy rozhrania pre iOS vyplývajúce zo sekcie 2.4.

Užívateľovi poskytuje možnosť nahlasovať dopravné priestupky spojené z nesprávnym parkovaním. Pre jednotlivý priestupok je v prvom kroku možné pridať jednu až tri fotografie priestupku. V ďalšom kroku kategóriu do ktorej patrí, popis a voliteľnú vzdialenosť, ak priestupok obsahuje nedodržanie stanovenej vzdialenosti. Vzdialenosť môže byť odmeraná pomocou rozšírenej reality alebo zadaná používateľom. Nasleduje výber polohy vozidla a záverečná rekapitulácia obsahujúca všetky vyplnené údaje. Po vyplnení všetkých potrebných informácii používateľ môže vytvoriť a odoslať alebo zrušiť pridávanie priestupku. Samotné vytvorenie je dostupné len pri dostupnosti internetu.

Po vytvorení priestupku sa záznam odošle na server, ktorý ho uloží do databázy. Uložený záznam je možné aktualizovať pridelenou osobou, teda správcom. Záznamy sú dostupné na webovej stránke dostupné pre správcu. Správca aktualizuje stav záznamu. Zmenený stav sa používateľovi aktualizuje vždy pri dostupnom internete pri zapnutí aplikácie alebo zobrazení záznamov. Prípadne potiahnutím nadol.

4.2 Cieľová skupina

Pre návrh používateľského rozhrania je potrebné určiť cieľovú skupinu používateľov. Cieľovou skupinou sú obce, mestá a správcovia verejných priestranstiev. Aplikácia sa zameriava na používateľov, ktorí sa cítia ohrození pri chôdzi po chodníku alebo po prechode pre chodcov. Prípadne im nie je umožnený voľný pohyb po mieste pre nich určenom. Každý občan nosí so sebou smartfón, ktorý ponúka rozsiahle možnosti využitia.

V súčasnosti je veľký nedostatok parkovacích miest a samosprávy nestíhajú budovať nové. Preto množstvo vodičov volí najjednoduchšiu variantu odstavenia vozidla kdekoľvek bez ohľadu na pravidlá. Správcovia a polícia nemôžu pokryť všetky miesta, kde sa vozidlá pohybujú alebo sú odstavené. Preto je vhodným riešením rozšíriť kontrolu o ob-

čana, ktorý môže nahlasovať priestupky. Tým by sa správcom alebo polícii vytvoril prehľad o miestach priestupkov, na ktoré sa môžu zamerať. Cieľom je priniesť rýchle a efektívne riešenie priestupkov, ktoré môže sledovať aj používateľ.

Aby si používateľ nainštaloval a používal mobilnú aplikáciu, musí byť jednoduchá na použitie a obsahovať moderné rozhranie. Od cieľového používateľa sa očakáva schopnosť základnej práce s chytrým telefónom. Konkrétne to sú minimálne znalosti obsluhy aplikácií fotoaparátu a máp. Používateľ bude využívať aplikáciu na nahlásenie nevhodne stojacich vozidiel. Ďalšou cieľovou skupinou používateľov sú šoféri vozidiel, ktorí často nemajú kde odstaviť svoje vozidlo alebo vidia iné vozidlá blokujúce viaceré parkovacie miesta. Predpokladá sa, že používatelia aplikácie budú nahlasovať odstavené vozidlá na verejnom priestranstve.

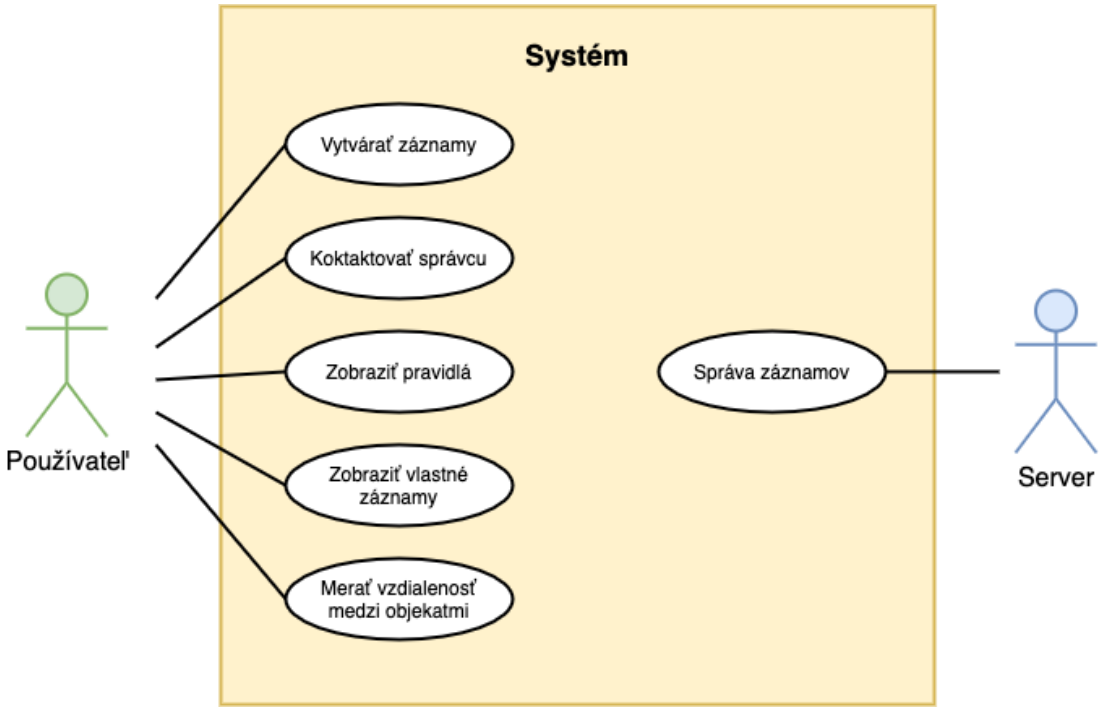
Používateľ by mal disponovať základnými pravidlami o cestnej premávke. Vekový rozsah používateľov je od 18 do 65 rokov, ktorí tvoria skupinu aktívnych šoférov a skupiny definované vyššie.

4.3 Prípady použitia

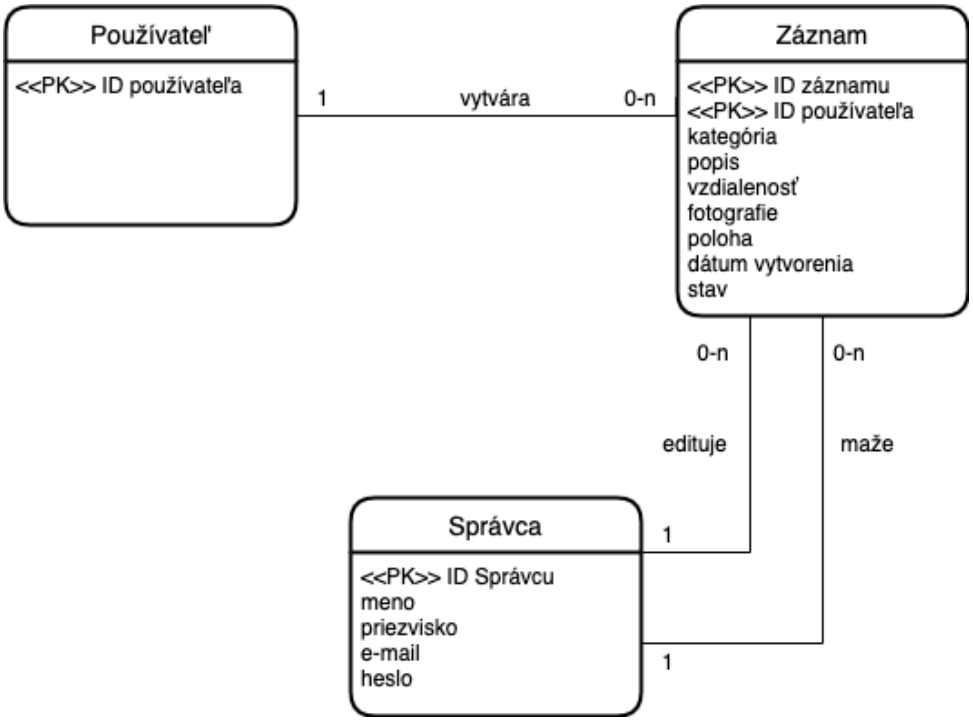
Diagram prípadov použitia sa nachádza na obrázku 4.1. Hlavnou funkciou aplikácie je vytváranie hlásení o nevhodne stojacich vozidlách. Tá obsahuje možnosť vybrať fotografie z galérie alebo zachytiť fotografie čo najjednoduchším spôsobom a pridať informácie o priestupku.

Na hlavnej obrazovke si používateľ zobrazuje všetky vlastné záznamy, ktoré vytvoril. Záznamy môže filtrovať podľa stavu. Ďalej používateľ môže kontaktovať správcu cez email a zobraziť pravidlá aplikácie.

Server zaznamenáva všetky hlásenia, ktoré používatelia nahlásia a ďalej ich spracúva správca systému. Záznamy sú zobrazené na webovej stránke pre správcu na ich editovanie.



Obr. 4.1: Diagram prípadov použitia



Obr. 4.2: ER diagram vzťahov databáze aplikácie

4.4 Návrh dátového modelu

Pre návrh konceptuálneho databázového modelu je využitý vzťahový model ERD[1] pre zobrazenie návrhu systému. Na obrázku 4.2 sú zobrazené modelové dáta a ich vzťahy. Popisuje uloženie dát v serverovej časti aplikácie. Podrobnejší popis vzťahov, dát a uloženie dát v zariadení je popísaný nižšie. Výsledkom návrhu dátového modelu je schéma databáze.

Používateľ

Uchováva používateľov unikátny identifikátor, ktorý určuje tvorcu záznamu. Vytvára záznamy a dostáva ich aktualizácie v prípade zmeny.

Záznam

Uchováva unikátne identifikátory záznamu a používateľa, kategóriu priestupku, popis, vzdialenosť, list URL[2] adries fotografií, polohu, dátum vytvorenia a stav. Poloha sa skladá zo zemepisných súradníc.

Správca

Uchováva unikátne identifikátory správcu, meno, priezvisko, e-mail a heslo. Slúži k editovaniu, mazaniu a zobrazovaniu všetkých záznamov.

[1] Entity-Relationship Diagram
[2] Uniform Resource Locator

Ukladanie dát v mobilnej aplikácii

Unikátny identifikátor používateľa je vytvorený po prvom spustení aplikácie a ostáva trvalo uložený v pamäti smartfónu. Ten sa prikladá ku každému vytvorenému záznamu. Tvar záznamu ostáva rovnaký ako v serverovej časti okrem identifikátora používateľa, ktorý už je uložený samostatne.

4.5 Návrh používateľského rozhrania

Pre tvorbu používateľského rozhrania boli využité zásady zo sekcie 2.4. Prvotný náčrt obrazoviek vznikol na papieri a neskôr na vytvorenie používateľského rozhrania bol použitý nástroj Figma[3]. Tento nástroj umožňuje vytvárať jednotlivé obrazovky aplikácie. Obrazovky je možné navzájom prepájať a vizualizáciou zobraziť ich vzájomné prepojenia. Jednotlivé štýly alebo komponenty obrazoviek je možné akokoľvek upravovať a kopírovať.

Na základe prieskumu konkurenčných aplikácií (Sekcia 3) spolu s prípadmi použitia (Obrázok 4.3) boli vytvorené prototypy obrazoviek iteratívnym spôsobom. Pri tvorbe používateľského rozhrania je nutné klásť dôraz na jednoduchosť a prehľadnosť aplikácie.

4.5.1 Prvá iterácia návrhu

V prvej iterácii boli prototypy vytvorené hlavne na základe prieskumu konkurenčných aplikácií. Hlavnú obrazovku (Obrázok 4.3 vľavo) tvoria náhľady hlásení vytvorených používateľom. Jednotlivé náhľady sú v zozname zoradené podľa ich času vytvorenia. Náhľad obsahuje zmenšenú fotografiu, názov a časť popisu hlásenia. Text popisu priestupku plynule prechádza z čiernej do svetlo šedej farby až zaniká. To používateľovi zvýrazňuje názov daného priestupku a zjednodušuje jeho čitateľnosť. Ďalej obrazovka obsahuje názov aplikácie, ikonu tlačidla nastavení a plávajúcu ikonu tlačidla na vytváranie hlásenia.

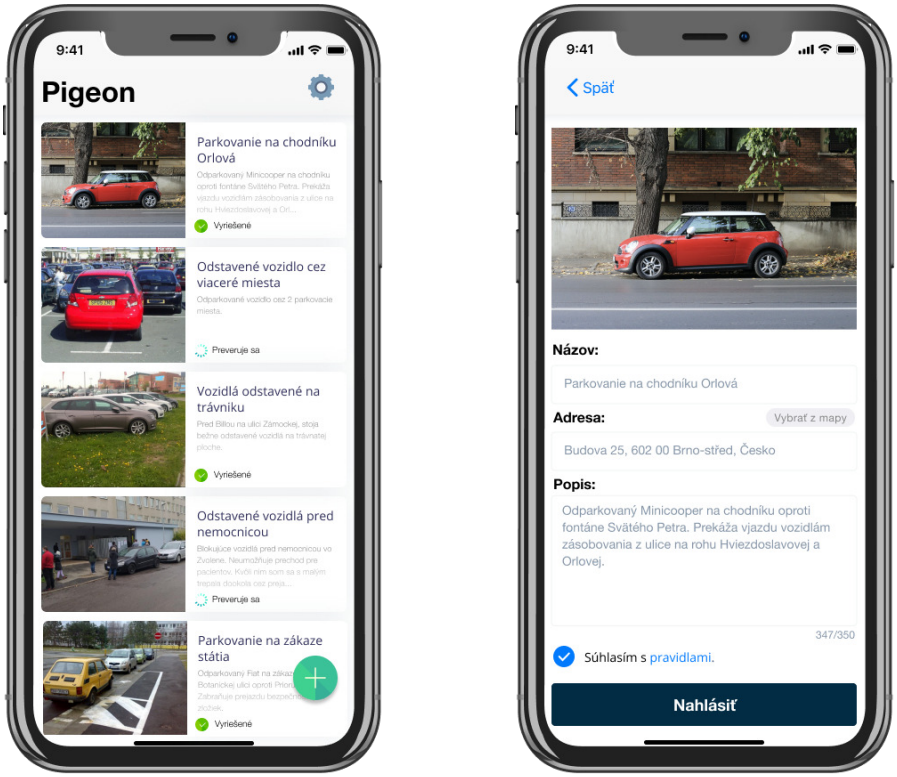
Tlačidlo na vytváranie hlásenia ponúka voľbu výberu fotografie z galérie alebo možnosť vytvorenia fotografie vstavanou aplikáciou fotoaparátu. Nasleduje formulár s povinnými poliami na vyplnenie názvu, adresy a popisu priestupku (Obrázok 4.3 vpravo). Adresu je možné vybrať z mapy pomocou GPS alebo manuálnym zadaním. Popis hlásenia je obmedzený na 350 znakov. Pred nahlásením podnetu je používateľ povinný zadať súhlas o porozumení pravidiel aplikácie.

Po otvorení náhľadu hlásenia sa zobrazí obrazovka detailu priestupku (Obrázok 4.4 vľavo). Obrazovka obsahuje fotografiu, názov, adresu a popis priestupku. Jednotlivé položky sa nedajú meniť. Používateľ si môže zobraziť len svoje priestupky. Taktiež si môže z hlavnej obrazovky otvoriť nastavenia (Obrázok 4.4 vpravo). Nastavenia ponúkajú možnosť povoliť alebo zakázať polohu. To vytvára používateľovi dojem, že má kontrolu nad aplikáciou. Ďalej nastavenia obsahujú verziu aplikácie a možnosť zobraziť pravidlá.

4.5.2 Druhá iterácia prototypu

Druhá iterácia vznikla na základe spätnej väzby na prvú iteráciu návrhu. Obsahovala chybné a nadbytočné prvky pre používateľa. Tlačidlo pridávania záznamu prekrývalo časť obsahu obrazovky a nebolo zrejmé, k čomu slúži. Nastavenia obsahovali správu povolení, ktorá je dostupná aj v nastaveniach iOS zariadenia. Pribudli možnosti filtrovať záznamy podľa stavu

[3] https://www.figma.com/



Obr. 4.3: Prototypy obrazoviek – zľava: Hlavná obrazovka, obrazovka pri vytváraní hlásenia

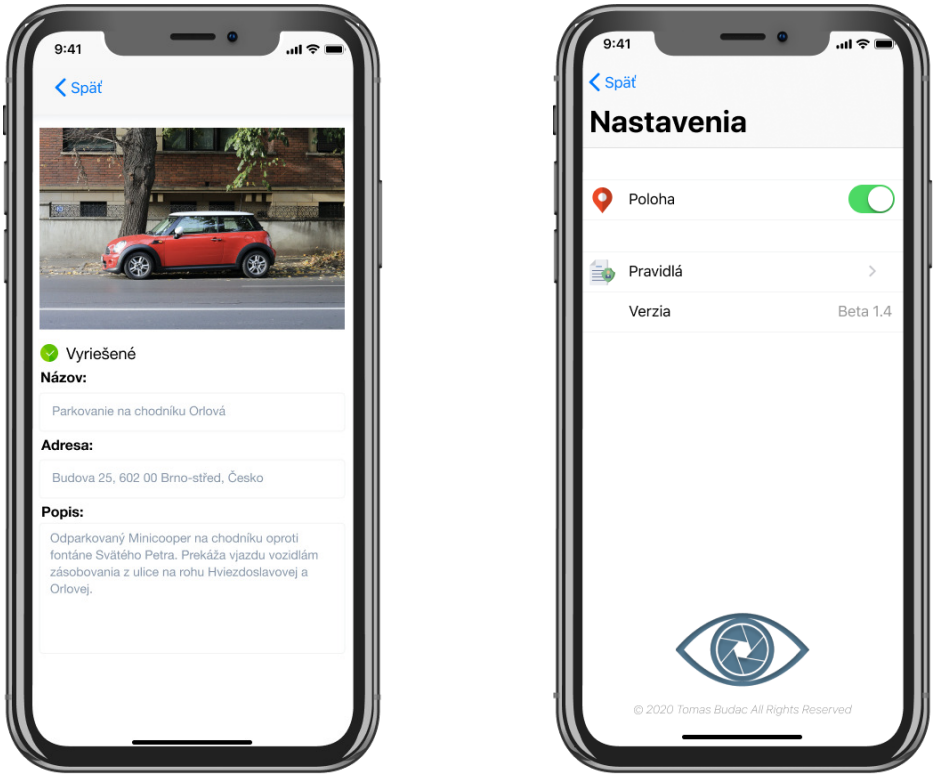
a kontaktovať správcu. Zmenila sa karta záznamu a pridávanie záznamu sa zmenilo z formuláru na pridávanie informácií sprievodcom po jednotlivých krokoch. Výsledky druhej iterácie návrhu sú použité pre výslednú implementáciu popísanú v kapitole 5, kde sú aj zobrazené jednotlivé obrazovky.

Hlavná obrazovka

Z hlavnej obrazovky (Obrázok 4.3 vľavo) bolo odobraté plávajúce tlačidlo pre pridávanie nového záznamu. Presunulo sa na miesto tlačidla nastavení (Obrázok 4.4 vľavo) a bola zvolená intuitívnejšia ikona vytvárania priestupku. Tlačidlo v pravom rohu vrchného navigačného panela je označovaná ako hlavná akcia obrazovky.[7]

Vľavo v navigačnom paneli pribudlo informatívne tlačidlo. Zobrazuje ponuku pre zobrazenie pravidiel alebo kontakt cez email (Obrázok 5.1 v pravo). V strede navigačného panelu je titulný popis obrazovky. Medzi panelom a zoznamom sa nachádza filter vo forme horizontálneho vyberača. Používateľovi ponúka lepší prehľad o stave záznamov. Rozdeľuje dlhý zoznam zmiešaných záznamov na viacero menších a tým zjednodušuje dohľadanie konkrétneho záznamu.

Karta záznamu je zväčšená na celú šírku obrazovky a pod ňu je presunutý popis (Obrázok 5.1 v strede). Stav záznamu je nahradený kategóriou záznamu, ktorý definuje oblasť do ktorej priestupok spadá.



Obr. 4.4: Prototypy obrazoviek – zľava: Obrazovka detailu hlásenia, obrazovka nastavení

Vytváranie priestupku

Pridávanie priestupku sa zmenilo na sprievodcu po jednotlivých krokoch (Obrázok 5.2). Používateľovi umožňuje nahrať viacero fotografií priestupku. Názov sa zmenil na kategóriu, čo prináša presnejšie určenie porušenia zákona. Popis ostal zachovaný. Vybratie polohy tvorí jeden krok pridávania.

Pre nedodržanie minimálnej vzdialenosti stanovenej zákonom sa pridala naviac podpora pre odmeranie vzdialenosti pomocou rozšírenej reality. V každom kroku vytvárania priestupku je používateľovi popísaná potrebná vec na vyplnenie. Svoj postup vytvárania vidí v ukazovateľovi postupu. Posledným krokom pridávania je rekapitulácia, ktorá zobrazuje všetky vyplnené informácie priestupku.

Obrazovka pravidiel a kontakt

Obrazovka pravidiel (Obrázok 5.4 pravo) ostala nezmenená. Definuje základné pravidlá pre používanie aplikácie a odkaz na zákony o cestnej premávke. Pribudla možnosť kontaktovať správcu pomocou emailu. Po zvolení možnosti kontaktu je otvorená obrazovka pre výber aplikácie zo zariadenia pre odosielanie emailov. Po výbere aplikácie sa otvorí nový email obsahujúci email správcu a predmet s používateľovým identifikátorom.

Kapitola 5

Implementácia

Samotná implementácia vznikala spolu s návrhom aplikácie v kapitole 4, kde sú popísané výsledky jednotlivých iterácií. Táto kapitola popisuje výslednú implementáciu, obmedzenia a riešenia problémov.

5.1 Všeobecné informácie

Pre implementáciu mobilnej aj serverovej časti aplikácie bolo použité grafické vývojové prostredie XCode vo verzii 12.4 a programovací jazyk Swift vezia 5.0. Pre vývoj slúžili zariadenia iPhone 8 s iOS verziou 14.4 a simulátor iOS zariadení. Cieľová verzia iOS je 14.0 a novšie, čo predstavuje 80% všetkých iOS zariadení. [3]

Pre lokálne uloženie dát v zariadení sa používa framework CoreData. Sieťová komunikáciu zo zariadenia na server zabezpečuje framework Alamofire. Serverová časť aplikácie je napísaná s využitím webového frameworku Vapor v programovacom jazyku Swift. Dáta z multipart sú na serveri ukladané pomocou frameworku Liquid. Ten dáta uloží a vracia ich URL adresu, kde sú uložené. Na prevádzanie dát a ich spracovanie zo silno typovaného jazyka Swift do objektovej reprezentácie dát v databáze PostgreSQL sa používa framework Fluent. FLUENT je rozšírením frameworku Vapor. Pre zobrazovanie dát z databáze na webe sa používa silno šablónový jazyk Leaf vytvorený na základe jazyka Swift. Umožňuje generovanie dynamických HTML stránok, kde sa používa zdrojová sada nástrojov Bootstrap.

Implementácia mobilnej aplikácie sa riadi návrhovým vzorom Model-View-ViewModel a jednotlivé vrstvy návrhového vzoru sú implementačne rozdelené do vlastných zložiek. Na základe využitia tohto návrhového vzoru ide jednoducho meniť jednotlivé časti samostatne bez potreby zmeny ostatných častí. V súčasnosti je to norma pri vývoji aplikácií prinášajúca mnoho výhod.

Alomofire

Alamofire je sieťová knižnica HTTP založená na programovacom jazyku Swift. Poskytuje rozhranie nad sieťovým balíkom Apple Foundation, ktorý zjednodušuje bežné sieťové úlohy. Medzi jeho hlavné funkcie patrí reťazenie požiadaviek a odpovedí, dekódovanie a kódovanie formátu JSON, autentifikácia a ďalšie.[1]

Metódy Alamofire:

- `AF.upload` – Nahrávanie súborov cez multipart, stream a rôzne dátové metódy.
- `AF.download` – Sťahovanie súborov alebo obnovenie sťahovania.

- `AF.request` – Ostatné požiadavky HTTP, ktoré nie sú spojené s prenosom súborov.

Všetky metódy sú globálne, takže nie je potrebné ich inštancovať pre použitie. Medzi základné prvky patria triedy a štruktúry ako dátové žiadosti a odpovede.

5.2 Mobilná aplikácia

Prvotne je potreba v `@main` definovať hlavnú obrazovku pri spustení aplikácie, z ktorej ďalej vychádzajú ďalšie obrazovky. Obrazovka je označovaná ako `View`, ktorá je súčasne aj protokolom. Slúži pre reprezentáciu údajov a logiky vychádzajúcej z view modelu s typom protokolu `ObservableObject` alebo `UIViewController`. View môže existovať samostatne aj bez nižšej vrstvy ako je napríklad ViewModel. Základnými stavebnými prvkami pre view sú `VStack` a `HStack`, ktoré určujú umiestnenie prvkov používateľského rozhrania. Medzi základné prvky patria napríklad `Text` pre prácu s textovou reprezentáciou a `Image` pre obrázky. Zobrazenie prvkov je možné meniť pomocou modifikátorov, ktoré formujú a upravujú prvky podľa zadaného modifikátoru.

Obrazovky sa ukladajú na zásobník obrazoviek. Vznikajú a zanikajú riadením udalostí typickým pre framework SwiftUI. Orientácia aplikácie je nastavená len na typ portrét.

5.2.1 Hlavná obrazovka so záznamami

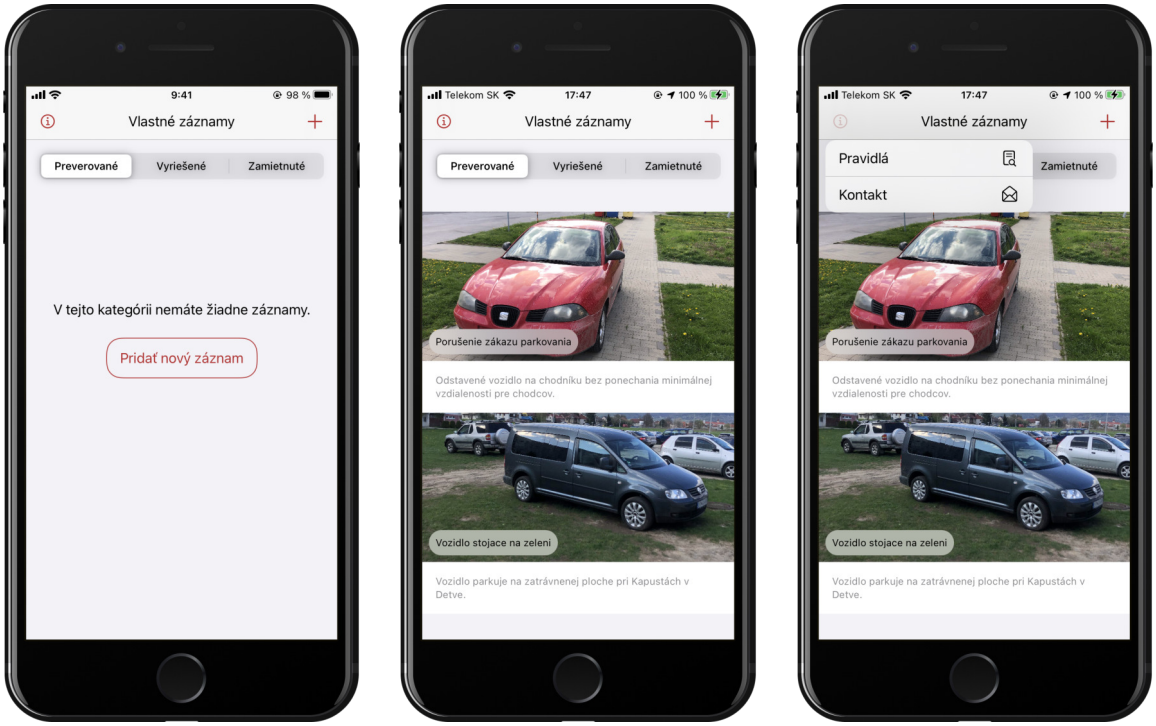
Hlavná obrazovka s jej viacerými možnosťami je zobrazená na obrázku 5.1. Ide o najdôležitejšiu časť aplikácie. Umožňuje používateľovi prehliadať si všetky vlastné záznamy, ktoré vytvoril. Pomocou filtra stavov si vie filtrovať záznamy podľa aktuálneho stavu riešenia. Dáta sú automaticky synchronizované so serverom. Pre používateľa je v hornej časti aplikácie v navigačnom paneli umiestnený názov obrazovky, ak by nevedel, kde sa v aplikácii nachádza. Súčasťou navigačného panela sú dve tlačidlá. Tlačidlo vľavo zobrazuje možnosti pre kontakt a pravidlá. Tlačidlo vpravo slúži na pridávanie nového záznamu. Ikony tlačidiel sú použité zo systémových ikôn systému iOS, tým sú užívateľovi už známe a nepotrebujú dodatočný popis. Celá implementácia obrazovky je vložená do `NavigationView`, ktorá vytvára navigačný zásobník. Navigačný zásobník umožňuje navigovať cez aplikáciu bez nutnosti definovania navigačných tlačidiel.

Zoznam záznamov a jeho aktualizovanie

Načítanie záznamov prebieha z lokálneho úložiska aplikácie v zariadení pomocou frameworku CoreData od spoločnosti Apple. Sú uchovávané ako perzistentné dáta, aby po vypnutí aplikácie ostali uložené v zariadení a boli dostupné aj bez pripojenia k internetu. Dáta je možné vymazať cez systémové nastavenia zariadenia pre aplikáciu.

Aktualizovanie stavu dát prebieha vždy pri zobrazení zoznamu záznamov alebo akcii potiahnutím dole pre opätovné aktualizovanie stavu zo serveru. Aktualizácia sa pošle prostredníctvom žiadosti `AF.request` pomocou metódy `GET` s adresou serveru a identifikátorom zariadenia vygenerovaného pri prvom štarte aplikácie. Odpoveďou serveru je zoznam všetkých používateľových záznamov v objekte obsahujúcom `recordUuid` a `status`. Ak neexistuje žiadny záznam, tak sa vracia prázdny zoznam.

Odpoveď je dekódovaná. Každý identifikátor záznamu uložený v zariadení, ktorý sa zhoduje s prijatým identifikátorom, tomu je aktualizovaný stav. Po aktualizácii stavu sa zmena uloží. Ak je stav aktuálny, zmena sa neukladá. V prípade príchodu neznámeho záznamu, je táto časť odpovede zahodená. Pre synchronizáciu a identifikáciu záznamov stačí



Obr. 5.1: Zobrazenie hlavnej obrazovky s možnosťami

unikátny identifikátor záznamu `recordUuid`. Nie je potreba posielať celý uložený záznam zo zariadenia, šetria sa tým prenesené dáta.

Každý záznam zoznamu je vytvorený ako tlačidlo presmerovania `NavigationLink` na obrazovku záznamu s parametrom typu `Record`. `Record` je model jednotlivého záznamu obsahujúci všetky dáta o zázname uloženom v pamäti zariadenia. Obrazovka je uložená na vrch navigačného zásobníka.

Filter

Filter stavov je implementovaný ako `Picker` s modifikátorom štýlu `SegmentedPickerStyle`. Obsahuje tri možné stavy pre záznam viditeľné na obrázku 5.1. Vybraný stav je zvýraznený a zoznam zodpovedajúci stavu je zobrazený. Je umiestnený medzi navigačným barom a zoznamom záznamov. Tým je vždy viditeľný a ponúka používateľovi prehľad o zvolenom filtri.

Navigačný panel

Panel umiestnený na vrchu obrazovky používa označenie navigačný panel. Je implementovaný ako `toolBar` obsahujúci tri menšie prvky `ToolbarItem` obsahujúce `Content` typu `View`. Typ `View` značí, že sa jedná o prvok používateľského rozhrania. Bez využitia `toolBar` nie je možné podľa dokumentácie pridávať ďalšie akcie od navigačného panelu okrem tlačidla späť.

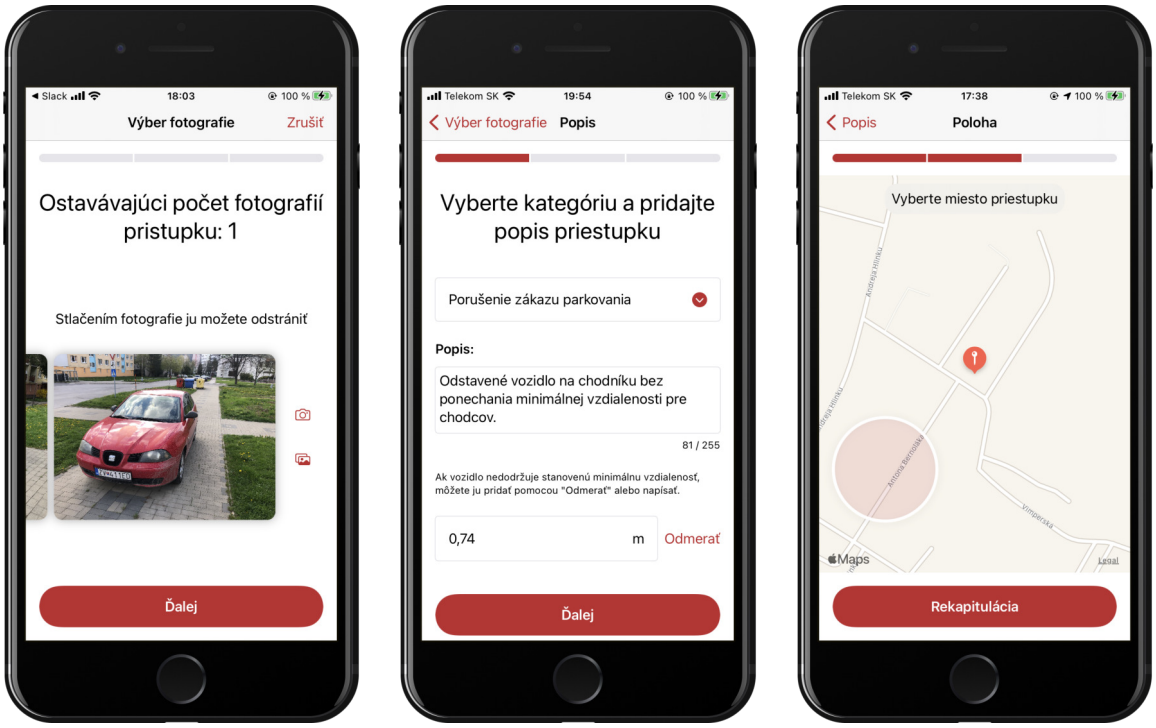
Umiestnenie v strede navigačného panelu je označované hlavné. Obsahuje pomenovanie obrazovky. Tlačidlo vľavo od pomenovania je sekundárna akcia. Zobrazuje ponuku

pre možnosť zobraziť pravidlá pre používanie aplikácie alebo možnosť kontaktovať správcu cez email.

Pre vytvorenie emailu sa používa `Controller`, ktorý rozširujú dva typy protokolov `UIViewController` a `MFMailComposeViewControllerDelegate`. Obidva protokoly sú potrebné pre vytváranie emailu do ktorého sa automaticky vloží preddefinovaný email správcu a predmet obsahujúci polovicu užívateľovho identifikátoru. Identifikátor je typu `UUID` s dĺžkou 32 znakov. Pre identifikáciu pomocou emailu je zvolená kvôli bezpečnosti len jeho polovica, aby sa neodosielal celý identifikátor uložený v databáze. Po kliknutí kontaktovať sa zobrazia možnosti aplikácií pre správu emailov v zariadení. Otvorenie pravidiel presmeruje používateľa na obrazovku pravidiel.

5.2.2 Obrazovky pre vytváranie priestupku

Obrazovky pre vytváranie priestupku (Obrázok 5.2) tvoria hlavnú funkcionalitu a účel mobilnej aplikácie pre nahlasovanie dopravných priestupkov spojených s nesprávnym parkovaním. Sprevádza používateľa pridávaním nového priestupku, inak nazývaného aj záznam. V jednotlivých krokoch vypĺňa potrebné informácie o priestupku. Ukazovateľ pokroku zobrazuje používateľovi ako ďaleko sa vo vytváraní nachádza. Zrušenie vytvárania nového záznamu môže z prvej obrazovky sprievodcu, kde vyberá zachytáva priestupok (Obrázok 5.2 vľavo) alebo z rekapitulácie. Na obrazovku rekapitulácie je používateľ presmerovaný po vyplnení povinných všetkých údajov z obrazovky výberu miesta záznamu. Medzi jednotlivými obrazovkami sa môže prepínať, ak sú všetky povinné údaje vyplnené v danej obrazovke.



Obr. 5.2: Obrazovky pre vytváranie nového priestupku

Výber fotografie

Používateľ má na výber zo zachytenia fotografie pomocou fotoaparátu zariadenia alebo výberom z galérie fotiek (Obrázok 5.2 vľavo). Pre pokračovanie musí vybrať aspoň jednu až tri fotografie priestupku. Pri pokuse vybrať alebo zachytiť fotografiu systém sám požiada o udelenie povolenia pre prácu s kamerou. Fotografie sú zobrazené v horizontálnom posuvnom zobrazení `ScrollView`. Stlačením obrázka sa otvorí modálne okno, ktoré umožňuje používateľovi odstrániť vybranú fotografiu.

Tlačidlá pre pridávanie fotiek sú dostupné v prípade, keď používateľ nevybral maximálny počet fotografií. Ak ešte nie je vybratá žiadna fotografia, vedľa ikôn sa zobrazí ich popis a tlačidlo na pokračovanie je neaktívne.

Pre zachytávanie fotografie sa využíva `UIImagePickerController`, ktorému je nastavený vstup `camera` pre otvorenie kamery. `Controller` je obsiahnutý vo frameworku SwiftUI a je podporovaný od verzie iOS 2.0. Jeho implementácia je jednoduchá a dobre popísaná v Apple dokumentácii. Pre zobrazovanie a zatváranie kamery je potrebný `coordinator`. `Coordinator` sa stará o `ViewController`, ktorý spravuje a zobrazuje, keď je potrebný.

Vyberanie fotografií používa `PHPicker` [14], ktorý je obsiahnutý vo frameworku PhotosUI. Podporuje verzie iOS 14.0 a novšie. Pre svoje použitie nepotrebuje systémové povolenie na zobrazenie galérie fotiek. Pri práci s prvotnou verziou `PHPicker` sa vyskytoval problém s chýbajúcim systémovým povolením. Pri novej aktualizácii frameworku sa problém odstránil.

Tlačidlo zrušiť v navigačnom paneli otvára modálne okno `Alert`, ktoré slúži používateľovi na potvrdenie vymazania záznamu bez jeho uloženia. Okno je možné zrušiť alebo potvrdiť. Ďalej obsahuje popis akcie tlačidla a výsledok jeho akcie. Tlačidlo na zrušenie je zvýraznené, aby sa predišlo nechcenému vymazaniu záznamu. Používateľovi to dáva možnosť potvrdiť akciu, ktorá nemôže byť vrátená aplikáciou.

Vyplnenie popisu

Používateľ vidí pokrok v ukazovateľovi postupu. Má možnosť vybrať si jednu z kategórií do ktorého priestupok patrí. Pridať popis priestupku obmedzeného na 255 znakov pričom minimálna dĺžka popisu je 5 znakov. Bez výberu kategórie alebo pridania popisu nemôže používateľ pokračovať. Voliteľným poľom na vyplnenie je vzdialenosť.

Výber kategórie je implementovaný ako `Picker` s modifikátorom štýlu `MenuPickerStyle`. Obsahuje zoznam kategórií priestupku. Pre pridávanie popisu je využitý `TextEditor` nepodporujúci náhradu textu `placeholder`. Pre jeho pridanie je možná implementácia pomocou `ZStack`, kde by prekrýval obsah `TextEditoru`. Toto riešenie je nepraktické a na rôznych iOS zariadeniach by bolo prekrytie nepresné. Ďalším riešením môže byť textové zadávacie pole `TextField`, ktoré obsahuje `placeholder`, ale je limitované na jeden riadok. Táto limitácia by používateľovi nedovoľovala zobraziť celý popis naraz. Z tohto dôvodu je pod popis pridané počítadlo znakov, ktoré má evokovať potrebu vyplnenia.

Voliteľným parametrom je vzdialenosť. Určuje vzdialenosť pre ponechanie potrebnej vzdialenosti napríklad pre chodcov, ak je vozidlo odstavené na chodníku. Používateľ môže zadať vzdialenosť alebo ju môže odmerať pomocou rozšírenej reality, popísanej v sekcii 5.2.3, s využitím frameworku ARKit.

Obrazovka výber polohy a obrazovka rekapitulácia

Pri otvorení obrazovky výberu polohy (Obrázok 5.2 vpravo) systém požiada o povolenie používať jeho polohu. Pre prácu s mapami a polohou používateľa je potrebný manažér polohy `CLLocationManager`. Tomu sú nastavené parametre pre požiadanie o prístup k polohe a presnosť určovania polohy na čo najpresnejšiu.

Po potvrdení používania polohy je mapa priblížená na používateľovu polohu, kde je zvýraznená krúžkom typickým pre zariadenia iOS. Ak nepotvrdí, mapa je zobrazená bez používateľovej polohy so zobrazením na celú krajinu nastavenú v základe na Slovenskú republiku. Polohu priestupku vyberá stlačením miesta na mapu. Túto akciu mapa rozoznáva pomocou `UITapGestureRecognizer`. Z mapy sa vyberie zvolený bod a uloží sa ako GPS súradnica typu `CLLocationCoordinate2D`. Používateľ môže zvoliť len jedno miesto. Ak je miesto už vybraté, ďalším stlačením iného miesta na mape sa predchádzajúce miesto odstráni z mapy a bod je pridaný na nové miesto. Vybratím miesta sa mapa vycentruje na stred s označeným miestom. Tento zvolený bod na mape je zobrazovaný ako `MKPointAnnotation`.

Po vybratí miesta je používateľovi umožnené pokračovať na obrazovku rekapitulácie. Rekapitulácia obsahuje všetky vyplnené údaje záznamu a adresu miesta vybrané z mapy. Ďalej má možnosť nahlásiť priestupok, vrátiť sa späť na predchádzajúce obrazovky na úpravu údajov alebo odstrániť priestupok. Tlačidlo nahlásenia je dostupné iba pri dostupnosti internetu. Ak používateľ nahlási priestupok, dáta sú uložené do pamäti zariadenia pomocou CoreData a odoslané na server. Dáta záznamu sú v tvare popísanom v návrhu dátového modelu v sekcii 4.4. Všetky pridané obrázky sú zmenšené pomocou kompresie `jpegData(compressionQuality: 0.40)`. Výsledkom kompresie je menšie potrebné miesto na ich ukladanie.

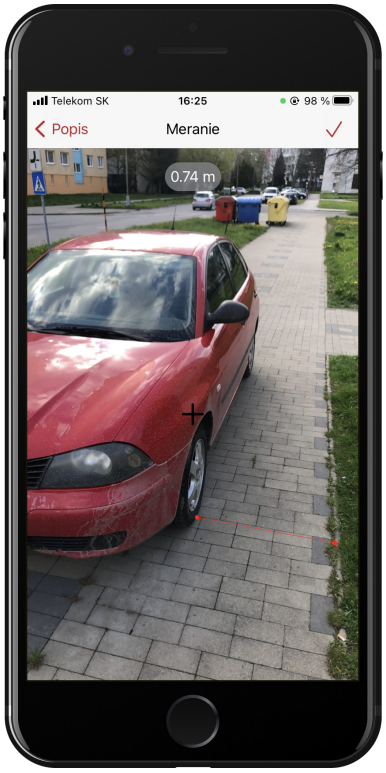
Dáta sú na server odosielané pomocou vlastnej triedy `Service`. Trieda najskôr vytvorí žiadosť na server obsahujúcu multipart pomocou `AF.upload`, kde časť multipart tvorí zoznam obrázkov. Jednotlivému obrázku je pridelený unikátny názov pre daný záznam. Žiadosť je odoslaná na adresu servera s identifikátormi používateľa a záznamu. V prípade úspechu server vráti zoznam URL adries obrázkov, kde sú zobraziteľné. V prípade úspechu vytvára novú žiadosť, inak sa pridávanie zruší.

Novo vytvorená žiadosť `AF.request` typu `POST` obsahuje kódované parametre objektu `RecordDTO`. `RecordDTO` predstavuje objekt pre prenos dát. Obsahuje všetky dáta `Record` naformátované do tvaru v akom ich server očakáva. Cieľová adresa obsahuje adresu serveru a identifikátor používateľa.

5.2.3 Meranie vzdialenosti pomocou rozšírenej reality

Pre prácu s rozšírenou realitou sa používa framwork ARKit popísaný v sekcii 2.10. Využíva fotoaparát zariadenia. Najskôr sa inicializuje `ARSCNView`, ktoré umožňuje pridávať 3D objekty do obrazovky kamery zariadenia. Pre správne používanie `ARSCNView` je potrebné nastaviť základnú scénu a `session`. `Session` detekuje všetky AR služby ako detekcia pohybu, analýza obrazu, svetelné podmienky a podobné. Na začiatku novej scény sa vždy vymažú všetky kotvy a detekcia sa vynuluje do začiatočného stavu.

Na zachytenie presnej pozície v skutočnom svete na akciu vyvolanú používateľom je potreba volať detekovanie objektov neustále. Preto sa metóda `detectObjects` neustále volá pri načítaní obrazu scény. V metóde sa volá metóda tranformácia vektora zo skutočného sveta uloženého v type objektu `CGPoint`, ktorý obsahuje x a y súradnicu pozície na obrazovke zariadenia. V metóde transformácie sa volá metóda `hitTest` obsiahnutá vo framworku ARKit.



Obr. 5.3: Obrazovka na meranie vzdialenosti v rozšírenej realite

Metóda `hitTest` sa snaží získať polohu objektu cez `CGPoint` spolu s metódou vysielania lúčov. V prípade úspechu vráti bod v priestore zobrazený v matici. Tento výsledok je ešte potrebný transformovať na 3D vektor, ktorým získavame skutočnú polohu daného bodu.

Keď používateľ začne dotyk na obraze z kamery, tak začne meranie vzdialenosti. Dotyk je detekovaný metódou `touchesBegan` obsiahnutej v protokole `UIViewController`. Reaguje na udalosť dotyku `UITouch`.

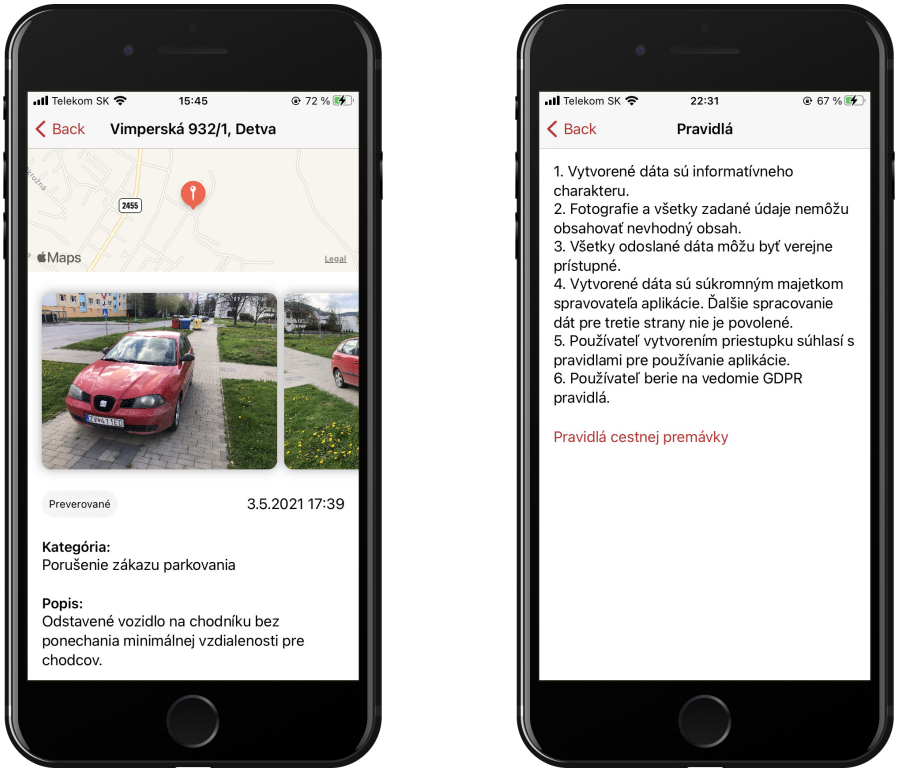
Samostatné meranie vytvára počiatočnú kotvu `SCNNode`, ktorej nastaví vzhľad červenej bodky (Obrázok 5.3) a vloží vytvorený objekt do scény. Od počiatočnej kotvy sa meria dĺžka vektora po koncovú kotvu, ktorá tvorí stred obrazovky. Ak používateľ prestane držať obraz z kamery, tak sa meranie preruší a uloží sa koncová kotva. Počiatočná a koncová kotva sú spojené úsečkou, ktorej dĺžka predstavuje skutočnú odmeranú dĺžku. Odmeraná vzdialenosť sa ukladá do poľa jej určenom popísanom vo vyplnení popisu 5.2.2.

Skutočný dosah vytvorenia počiatočnej kotvy pre meranie je dva až tri metre od zariadenia za dobrých svetelných podmienok. Keď je počiatočná kotva nastavená, používateľ sa môže pohybovať po priestore a zisťovať vzdialenosť od kotvy. Scéna po ktorej sa používateľ môže pohybovať nie je obmedzená.

5.2.4 Obrazovky detailu priestupku a pravidlá

Obrazovka detailu (Obrázok 5.4 vľavo) záznamu obsahuje súhrnné informácie o nahlásenom priestupku. Informácie o priestupku sú uložené v pamäti zariadenia. Obrazovka zobrazuje polohu, obrázky, stav, dátum a čas vytvoreniam, kategóriu, popis a vzdialenosť priestupku. Vzdialenosť je zobrazená, ak bola vyplnená pri vytváraní priestupku. Zobrazenie obrázkov

je možné v horizontálnom posuvnom riadku. Navigačný panel obsahuje adresu priestupku ako názov obrazovky a tlačidlo späť na hlavnú obrazovku.



Obr. 5.4: Obrazovky detailu záznamu a pravidiel zľava

Obrazovka pravidiel (Obrázok 5.4 vpravo) obsahuje súhrn pravidiel pre používanie aplikácie a odkaz na webovú stránku platných pravidiel cestnej premávky.

5.3 Serverová časť

Úlohou serverovej aplikácie je spracovanie požiadaviek od klienta a zobrazovanie dát pre správcu. Je implementovaná vo frameworku Vapor verzii 4.0. Je rozšírený o framworky Fluent, Liquid, MuiltipartKit a Leaf. Využitie jednotlivých frameworkov v aplikácii je opísaný nižšie. Pre správu kontajneru databáze sa používa otvorený softvér Docker.

5.3.1 Súborová štruktúra

Zložka `Public` v sebe obsahuje dve zložky `RecordImages` a `Styles`. `RecordImages` v sebe obsahuje všetky nahraté obrázky priestupkov rozdelené podľa identifikátorov používateľa a priestupku. `Styles` obsahuje kaskádové štýly pre zobrazenie webového rozhrania. Zložka `Resources` je určená pre webové rozhranie stránok napísané v značkovacom jazyku HTML s využitím frameworkov Leaf a Bootstrap. `Sources` obsahuje zdrojové súbory v programovacom jazyku Swift. Ďalej obsahuje priečinky najvyššej úrovne, `App` a `Run`, ktoré vytvárajú balíky modulov pre spúšťanie aplikácie deklarované v súbore `Package` typu Swift. V `Package` súbore sú definované všetky potrebné závislosti pre vytváranie spustiteľnej aplikácie. Musí byť uložená v základnej zložke projektu. V zložke `App` sa nachádza logika aplikácie.

Popis zložiek a súborov v zložke `App`:

- `Controllers` – Obsahuje kolekcie metód, ktoré prijímajú a vracajú odpovede.
- `Migrations` – Obsahuje schému databázy, ktorá určuje aké parametre sú povinné a nepovinné. Používa sa vždy pri aktualizáciách alebo v zmenách databáze s využitím frameworku Fluent.
- `Models` – Sú v ňom uložené základné modely dát. V implementovanej aplikácii sa vytvára model pre záznam `RecordModel`, tým definuje entitu uloženú v databáze PostreSQL.
- `configure.swift` – Konfiguruje aplikáciu pre jej spustenie. Ďalej registruje používané služby ako databázu, koncové body, miesto ukladania dát a podobne.
- `routes.swift` – Definuje koncové body a ich spracovanie.

Súbor `main` typu Swift vytvára a spúšťa nakonfigurovanú inštanciu aplikácie. Zložka `Tests` obsahuje základné jednotkové testy aplikácie.[22]

5.3.2 Koncové body

Koncové body sú definované v súbore `routes.swift`. Spracovávajú požiadavky odoslané pomocou protokolu HTTP a jej metódami. Z mobilnej aplikácie popísanej v sekcii 5.2 sú odosielané požiadavky na vytváranie záznamu v databáze, ukladanie dát a získavanie informácií o zázname z databáze. Všetky žiadosti server spracúva asynchrónne. Tým neobmedzuje jeho dostupnosť na spracovanie ďalších požiadaviek.

Požiadavky typu `POST` slúžia na zapisovanie dát do databáze alebo ukladanie dát z multipart. Požiadavky na zapisovanie nových záznamov do databáze obsahujú parametre. Parametre predstavujú reprezentáciu dát, ktoré majú byť uložené. Z mobilnej aplikácie sa odosielajú v rovnakom tvare modelu, aké server očakáva. Inak nedokáže požiadavku spracovať. Príkladom vytvárania nového záznamu databáze je nahrávanie nového priestupku (Výpis 5.1).

```
app.post("uploadRecord") { req -> EventLoopFuture<RecordModel> in
    let record = try req.content.decode(RecordModel.self)
    return record.create(on: req.db)
        .map { record }
}
```

Výpis 5.1: Ukážka spracovania požiadavky typu `POST`

Požiadavka vytvára generický typ `EventLoopFuture` pre daný model `RecordModel`. Pripojené parametre je potrebné dekódovať daným modelom. Model má presne definované typy pre parametre. Ak sa dekódovanie nepodarí, vráti sa neúspech odosielateľovi požiadavky. Ak sa podarí, vytvorí sa nový záznam v databáze s dátami z parametrov.

Požiadavky typu `GET` slúžia na získavanie údajov z databáze alebo na vytvorenie a presmerovanie na webové rozhranie. Pre získavanie údajov sa využíva metóda `query` nad databázou záznamov daného typu. Metóde sú pridané modifikátory pre filtrovanie záznamov nad databázou. Výsledok filtrovania je spracovaný a vrátení odosielateľovi požiadavku. V prípade neúspechu je vrátená chyba.

5.3.3 Ukladanie dát do pamäte

Pre ukladanie dát je potrebné prvotne definovať miesto úložiska v konfiguračnom súbore. Úložisko je definované ako zložka `Public` pomocou frameworku Liquid. Ďalej adresu URL pre prístup k uloženým dátam, konkrétnu zložku na ukladanie dát a typ úložiska. Typ je nastavený ako lokálny, keďže server tvorí aj úložisko. Maximálna veľkosť dát v jednej požiadavky je nastavený na 10 MB.

Ukladanie dát je implementované pre možnosť nahrávať obrázky na disk, kde sa nachádza aj server. Je možnosť ukladať obrázky do databázy, ale ich načítanie by vyžadovalo jej opätovné prehľadávanie. Ukladanie na disk je praktickejšie a obrázky môžu byť zdieľané URL adresou.

Jednotlivé dáta z požiadavky odosielateľa z časti multipart sú dekódované vo formáte zoznamu dvojíc `[String:Data]`. Potom sa skontroluje správnosť parametrov požiadavky a prijatých dát. Prvok zoznamu je dvojica názvu obrázku a dáta obrázku. Každý prvok zoznamu je nahratý pomocou metódy `upload` obsiahnutej vo frameworku Liquid. Metóda vráti URL adresu obrázku. Zoznam vytvorených URL adries obrázkov je späť odoslaný odosielateľovi požiadavku. URL adresu je možné zadať do webového prehliadača a obrázok bude zobrazený. V prípade neúspechu dáta nebudú uložené a je vrátený neúspech.

5.3.4 Webové rozhranie

Pre vytvorenie webového rozhrania stránky sa používa framework Leaf. Pre vytvorenie používa metódu `render`, ktorej vloží názov súboru stránky a kontext. Metódu zavolá spracovanie požiadavky v `routes`. Súbor stránky je napísaný v jazyku HTML a používa framework Leaf na dynamické generovanie stránky. Na úpravu vzhľadu rozhrania sa používa zdrojová sada nástrojov Bootstrap a kaskádové štýly.

Kontext predáva stránke dáta, ktoré slúžia na dynamické vytváranie stránky. Sú to napríklad dáta záznamov, ktoré dynamicky vytvárajú tabuľku záznamov. Kliknutím na záznam je používateľ presmerovaný na detail záznamu.

Stránka detailu záznamu obsahuje všetky obrázky, informácie a možnosť zmeniť stav priestupku správcom. Taktiež obsahuje navigačný panel obsahujúci tlačidlo, ktoré presmeruje používateľa na zoznam záznamov. Zmena stavu je v databáze vykonaná hneď.

Kapitola 6

Testovanie

Najčastejšou činnosťou behom celého vývojového procesu vývoja obidvoch aplikácií bolo testovanie. Testy mobilnej aplikácie počas vývoja prebiehali v grafickom vývojovom prostredí XCode za použitia mobilného simulátoru iOS zariadení a na fyzickom zariadení iPhone 8. Serverová časť bola testovaná jednoduchým skriptom obsahujúcim požiadavky, softvérovou aplikáciou RESTed[1] a z mobilnej aplikácie. Po jednotlivých iteráciách vývoja bola vytvorená aplikácia testovaná s používateľmi na skutočných zariadeniach iPhone. Po dokončení implementácie aplikácií, prebiehalo skutočné testovanie v meste Detva. V tejto kapitole sú popísané jednotlivé časti testovania.

6.1 Testovanie mobilnej aplikácie

Pre testovanie mobilnej aplikácie bol priebežne používaný simulátor iOS zariadení a náhľad podporovaný frameworkom SwiftUI. Umožňovalo to testovanie funkcionality jednotlivých častí a vizualizáciu používateľského rozhrania. Simulátor umožnil zobrazovať rôzne iOS zariadenia a tým rôzne veľkosti obrazoviek. Testovanie kamery a rozšírenej reality prebiehalo na fyzickom zariadení iPhone 8. Simulátor neobsahuje kameru.

6.2 Prvá iterácia testovania

Cieľom prvej iterácie bolo vytvoriť prototyp aplikácie podľa prieskumu už existujúcich riešení. Používateľské rozhranie prvej iterácie je popísané v sekcii 4.5.1. Aplikácia bola implementovaná podľa predom vytvorených návrhov, čo sa ukázalo ako dobrý nápad. Z tohto dôvodu som mohol predom priblížiť používateľovi ďalšie časti aplikácie, ktoré ešte neboli implementované alebo napojené na server. Aplikácia počas prvej iterácie ešte nemala implementovanú serverovú časť. Všetky dáta boli ukladané len do zariadenia.

Aplikácia bola nainštalovaná 10 testerom vo veku od 18 do 50 rokov. Päť testerom bola predom prezentovaná aplikácia o jej možnostiach a využitiach spolu s návrhom používateľského rozhrania. Ďalším piatim testerom nebola aplikácia predstavená. Následne testeri boli požiadaní o vykonanie akcií predom pripraveného scenára. Scenár bol zameraný na postupné prejdenie celej funkcionality aplikácie. Pripomienky a reakcie boli zapisované a následne analyzované.

Zistené problémy:

[1] RESTed – https://apps.apple.com/us/app/rested-simple-http-requests/id421879749?mt=12

- Nepochopenie účelu aplikácie – testeri, ktorím predom nebol vysvetlený účel aplikácie ho nepochopili. Bolo to spôsobené nedostatočnými popismi pre pridávanie nového priestupku.
- Názov nepribližoval priestupok dostatočne.
- Nastavenia neobsahovali dostatok možností.
- Tlačidlo pridávania záznamu prekrývalo zoznam priestupkov.
- Nezobrazuje sa žiadne potvrdenie pre zrušenie pridávania priestupku.
- Ukážka priestupku je veľmi malá a neprehľadná.
- Chýbajúce alebo nevhodné názvy pre obrazovky.

6.3 Druhá iterácia testovania

Cieľ druhej iterácie testovania predstavuje opravenie nedostatkov z predchádzajúcej iterácie a test funkcionality mobilnej aplikácie so serverom. Na základe prvej iterácie bol vytvorený nový návrh používateľského rozhrania.

Vytvorený návrh bol prezentovaný piatim testerom. Testeri pripomienkovali napríklad nedostatky:

- Chýbajúce filtrovanie priestupkov.
- Ukážka priestupku neobsahovala kategóriu.
- Nemožnosť kontaktovať správcu.

Tieto nedostatky boli odstránené a sú obsiahnuté v druhej iterácii návrhu 4.5.1. Výsledkom druhej iterácie návrhu sa stalo aj výsledne používateľské rozhranie zobrazené v sekcii 5.2.

Pri implementácii druhej iterácie návrhu bolo potrebné testovať mobilnú aj serverovú aplikáciu zvlášť a následne spolu pre vzájomnú komunikáciu. Na testovanie mobilnej aplikácie sa okrem simulátoru muselo použiť aj fyzické zariadenie iPhone 8. Fyzické zariadenie slúžilo na testovanie kamery a rozšírenej reality.

Testovanie rozšírenej reality počas implementácie prebiehalo v interiéri a exteriéri. Na testovanie merania v rozšírenej realite boli použitý dvaja testeri. Skúšali dosah za rôznych svetelných podmienok na zariadeniach iPhone 8 a iPhone 12 mini. Testovanie odhalilo nedostatočne výrazné zobrazenie kvót a čiary spájajúcej kvóty.

Testovanie samostatnej serverovej časti prebiehalo odosielaním HTTP požiadaviek na server jednoduchým scriptom alebo softvérovou aplikáciou RESTed. Script obsahoval základné HTTP požiadavky. Zložitejšie požiadavky obsahujúce boli odosielané z mobilného zariadenia. Výsledkom bolo odhalenie chýbajúcich koncových bodov a nesprávne ukladanie dát. Ďalšie testovanie komunikácie testermi je popísané v ďalšej sekcii.

6.3.1 Záverečné testovanie v meste Detva

Cieľom testovania je odskúšať aplikáciu v bežnom používaní používateľmi. Pre toto testovanie boli vytvorené ukážkové dáta v mobilnej aplikácii ako vzor na prezentáciu aplikácie.

Priestupky sa nachádzali na skutočných miestach mesta Detva. Pre testovanie bolo vybratých päť testerov bez znalosti problematiky vývoja aplikácií. Predom boli oboznámení, že sa testuje samotná aplikácia a nie ich vedomosti. Počas testovania boli zaznamenávané ich pripomienky alebo problémy.

Vytváranie nového priestupku

Keď používateľ chce vytvoriť nový priestupok, musí stlačiť tlačidlo plus z hlavnej obrazovky. Následne vybrať alebo zachytiť fotografiu a pokračovať ďalej. Ďalej vybrať kategóriu, vyplniť popis a možnosť vyplniť alebo odmerať vzdialenosť. Potom vybrať miesto a pokračovať na rekapituláciu. V rekapitulácii stlačiť tlačidlo nahlásiť.

S touto úlohou nemali testeri problém. Jeden tester pripomienkoval chýbajúci zástupný text pre popis. Kladne hodnotili možnosť odmerať vzdialenosť pomocou rozšírenej reality a automatické vyplnenie políčka vzdialenosti po odmeraní.

Prechádzanie vytvorených priestupkov

Zobrazenie zoznamu priestupkov používateľ vidí na hlavnej obrazovke. Priestupky môže filtrovať podľa stavu pomocou vyberača. Stlačením na priestupok sa mu otvorí detail priestupku so všetkými zadanými informáciami.

Testeri pripomienkovali šedý okraj textových polí v detaile priestupku. Mysleli, že ho môžu dodatočne upravovať. Taktiež chýbala adresa priestupku a mapa neposkytovala túto informáciu. Naopak sa im páčil zoznam priestupkov a jeho možnosť obnoviť potiahnutím dolu. Inak boli testeri spokojní.

Zobrazenie pravidiel a kontakt

Pre otvorenie pravidiel alebo možnosť kontaktu, používateľ musí stlačiť tlačidlo ikony informácie v navigačnom paneli vľavo. Zobrazia sa mu na výber možnosti pre zobrazenie pravidiel alebo kontaktu. Stlačením pravidiel sa zobrazí obrazovka pravidiel. Pri možnosti kontaktovať sa otvorí obrazovka s aplikáciami zariadenia pre správu emailov. Po vybratí aplikácie sa otvorí nový email s vyplneným emailom správcu a predmetom obsahujúcim stručný popis spolu s identifikátor používateľa.

Túto funkcionalitu testeri hodnotili kladne. Páčila sa im možnosť kontaktovať správcu rovno cez predom pripravený email.

Monitorovanie a správa dát na serveri

Testovanie pridávania nových priestupkov bolo kontrolované na webovom rozhraní, či všetky dáta sú prijaté a správne. Nahraté dáta si správca zobrazuje pomocou webového rozhrania. Webová stánka obsahuje tabuľku všetkých priestupkov hlavnými údajmi. Po kliknutí na riadok tabuľky sa správcovi otvorí nová stránka zodpovedajúca priestupku. Stránka priestupku obsahuje všetky údaje a obrázky priestupku. Ďalej môže meniť jej stav alebo sa vrátiť späť na tabuľku priestupkov.

K webovému rozhraniu testeri nemali žiadne pripomienky. Ocenili jednoduchosť a prehľadnosť. Všetky vytvorené záznamy z mobilnej aplikácie boli dostupné aj na webe. Tým sa overila správnosť ukladania a zobrazovania dát. Zmenený stav priestupku z webu sa zobrazil aj v mobilnej aplikácii.

Meno	Funkčnosť	Vzhľad	Jednoduchosť	Myšlienka	Odporučenie?
Lukáš	9/10	8/10	10/10	8/10	áno
Viktória	10/10	9/10	10/10	9/10	áno
Ján	10/10	9/10	9/10	10/10	áno
Andrej	9/10	10/10	9/10	9/10	áno
Norbert	9/10	8/10	9/10	9/10	áno

Tabuľka 6.1: Tabuľka hodnotenia aplikácií testermi

Vyhodnotenie testovania

Na konci testovania testeri dostali dotazník (Tabuľka 6.1) na vyplnenie. Dotazník obsahoval hodnotenie funkcionalít a dojmov z aplikácií na škále od 1 do 10, kde 10 predstavuje najlepšie hodnotenie. Proces testovania prebehol bez problémov a s testermi sa mi spolupracovalo príjemne. Celkové hodnotenie testerov dopadlo kladne a ocenili myšlienku aplikácií.

Kapitola 7

Záver

Cieľom tejto práce bolo popísať problematiku vývoja aplikácie pre operačný systém iOS a serverovej aplikácie s webovým rozhraním. Na začiatok bolo potrebné navrhnúť mobilnú aplikáciu umožňujúcu nahlasovať dopravné priestupky. V prvej iterácii bol návrh implementovaný a testovaný. Na základe výsledkov prvej iterácie vznikla druhá iterácia obsahujúca rovnaký postup. Následne po iteráciách prebiehalo testovanie. Výsledkom je plne funkčný systém poskytujúci možnosť nahlasovania priestupkov z mobilnej aplikácie a ich spracovanie vo webovom rozhraní. Pre implementáciu je využitý moderný programovací Swift dbajúci na bezpečnosť. Prvý návrh aplikácie vychádza z analýzy už existujúcich riešení.

Informácie potrebné pre návrh a implementáciu som získal z konzultácií s vedúcim bakalárskej práce, z dokumentácií k použitým programovacím jazykom a manuálov technológií, kníh rozoberajúcich problematiku vývoja a technologických článkov.

Pri tvorbe aplikácií bol kladený dôraz na dodržanie celého rozsahu zadania práce. Výsledný systém spolu s prílohami spĺňa všetky body zadania. Zadanie bolo rozšírené o meranie vzdialenosti pomocou rozšírenej reality a serverovou aplikáciou s webovým rozhraním.

Počas tvorby systému bola nadviazaná spolupráca s mestskou políciou mesta Detva. Počas vývoja pandemické opatrenia nedovoľovali nasadiť systém do skutočnej prevádzky a spolupráca sa prerušila. Následne bola kontaktovaná správa parkovania v meste Brno, kde odpoveď trvala niekoľko mesiacov a spolupráca ostáva otvorená.

Systém funguje spoľahlivo a rýchlo. Do budúcnosti je tu možnosť na ďalšie rozšírenia hlavne pre webové rozhranie, kde chýba filtrácia záznamov a zobrazovanie všetkých zmien s dátumom. Ďalšie rozšírenia a nasadenie do prevádzky závisí od spoluprác. Ďalej je možnosť rozšíriť systém o nahlasovanie akýchkoľvek závad na verejnom priestranstve, čím by sa rozšírila používateľská skupina.

Literatúra

[1] Alamofire Software Foundation: *Alamofire Elegant Networking in Swift*. [Online; navštívené 2021-04-16].
URL http://alamofire.github.io/Alamofire/

[2] Ali Madooei: *Client-Server Application*. [Online; navštívené 2021-04-28].
URL https://madooei.github.io/cs421_sp20_homepage/client-server-app/

[3] Apple Inc.: *App Store - iOS and iPadOS Usage*. [Online; navštívené 2021-04-19].
URL https://developer.apple.com/support/app-store/

[4] Apple Inc.: *Cameras and Media Capture*. [Online; navštívené 2021-05-08].
URL https://developer.apple.com/documentation/avfoundation/cameras_and_media_capture

[5] Apple Inc.: *Framework - SwiftUI*. [Online; navštívené 2021-01-12].
URL https://developer.apple.com/documentation/swiftui

[6] Apple Inc.: *Introducing ARKit: Augmented Reality for iOS*. [Online; navštívené 2021-04-10].
URL https://developer.apple.com/videos/play/wwdc2017/602/

[7] Apple Inc.: *iOS Design Themes - Human Interface Guidelines*. [Online; navštívené 2021-01-07].
URL https://developer.apple.com/design/human-interface-guidelines/ios/overview/themes/

[8] Apple Inc.: *Swift*. [Online; navštívené 2020-12-25].
URL https://developer.apple.com/swift/

[9] Apple Inc.: *Understanding World Tracking*. [Online; navštívené 2021-04-10].
URL https://developer.apple.com/documentation/arkit/configuration_objects/understanding_world_tracking

[10] Daniel Eggert: *How Your Camera Works*. [Online; navštívené 2021-05-07].
URL https://www.objc.io/issues/21-camera-and-photos/how-your-camera-works/

[11] DigitalThinkerHelp: *What is Client Server Architecture: Diagram, Types, Examples, Components*. [Online; navštívené 2021-04-28].
URL https://digitalthinkerhelp.com/what-is-client-server-architecture-diagram-types-examples-components/

[12] Josh Biggs: *Top 7 Programming Languages for iPhone App Development*. 2009-04-21 [Online; navštívené 2021-01-08].
URL https://www.meldium.com/top-7-programming-languages-for-iphone-app-development/

[13] Paul Hudson: *Answering the big question: should you learn SwiftUI, UIKit, or both?* 2019-09-18 [Online; navštívené 2021-01-18].
URL https://www.hackingwithswift.com/quick-start/swiftui/answering-the-big-question-should-you-learn-swiftui-uikit-or-both

[14] Prafulla Singh: *SwiftUI: How to use PHPicker(PhotosUI) to select Image from Library?* [Online; navštívené 2021-04-19].
URL https://blog.devgenius.io/swiftui-how-to-use-phpicker-photosui-to-select-image-from-library-5b74885720ec

[15] Quiller Media, Inc.: *ARKit*. [Online; navštívené 2021-04-10].
URL https://appleinsider.com/inside/arkit

[16] Red Hat, Inc.: *What is a REST API?* [Online; navštívené 2021-04-08].
URL https://www.redhat.com/en/topics/api/what-is-a-rest-api

[17] Robert Mejia: *Declarative and Imperative Programming using SwiftUI and UIKit*. 2019-11-12 [Online; navštívené 2021-05-07].
URL https://medium.com/flawless-app-stories/declarative-and-imperative-programming-using-swiftui-and-uikit-c91f1f104252

[18] Sophie Clark: *An iOS Developers journey into Server-Side Swift using Vapor*. [Online; navštívené 2021-04-07].
URL https://medium.com/comparethemarket/an-ios-developers-journey-into-vapor-c8ddd7cbef07

[19] Subhash Chandra Yadav, S. K. S.: *Introduction to Client/Server Computing*. New Age International Publisher, 2009, ISBN 9788122428612.

[20] The-Crankshaft Publishing: *Image Acquisition (Introduction to Video and Image Processing) Part 1*. [Online; navštívené 2021-05-07].
URL
http://what-when-how.com/introduction-to-video-and-image-processing/image-acquisition-introduction-to-video-and-image-processing-part-1/

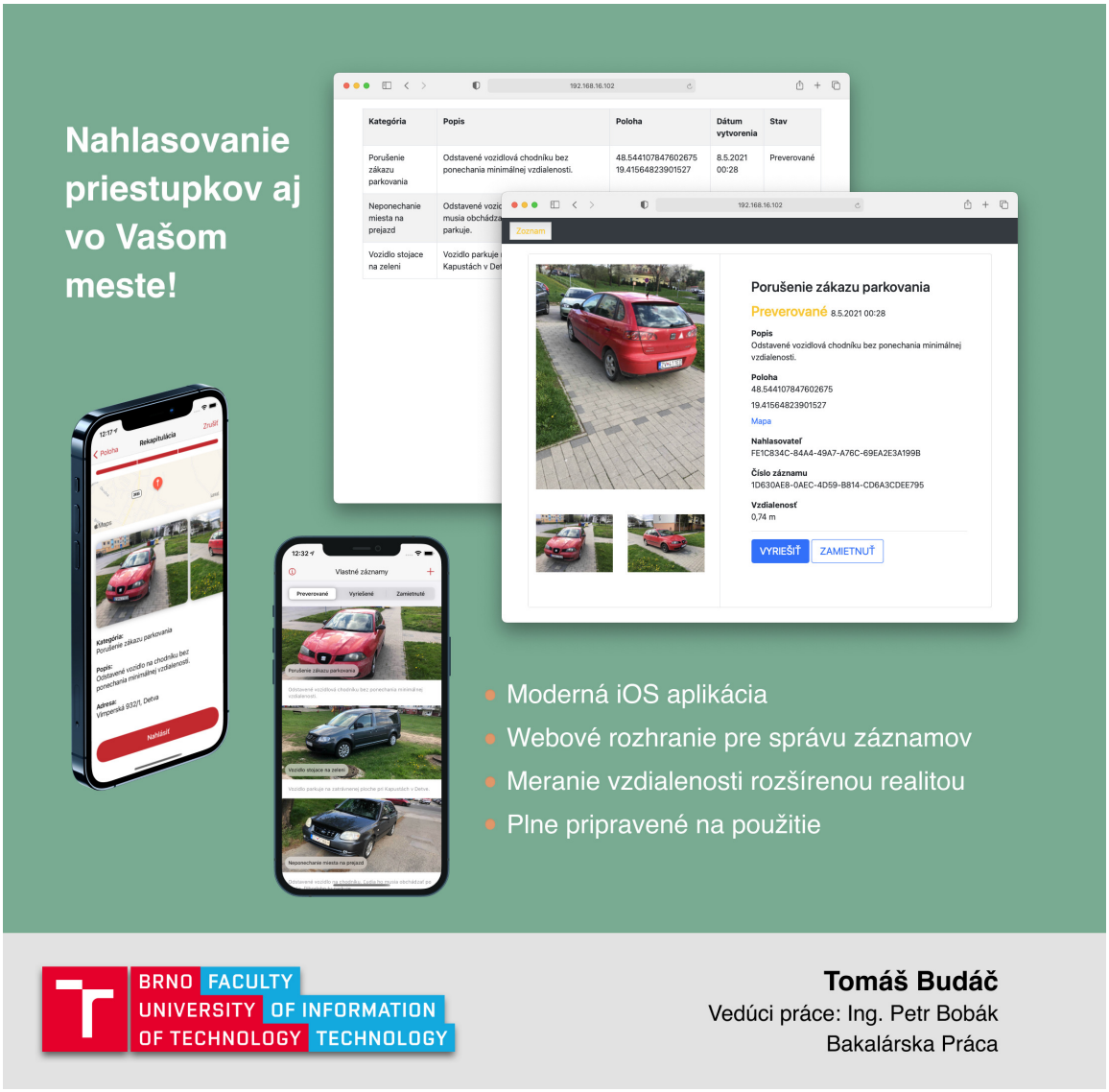
[21] U.S. General Services Administration: *User Interface Design Methods*. [Online; navštívené 2021-05-07].
URL https://www.usability.gov/how-to-and-tools/methods/user-interface-design/index.html

[22] Vapor Community: *Vapor Docs*. [Online; navštívené 2021-04-20].
URL https://docs.vapor.codes/4.0/

[23] Václav Dajbych: *MVVM: Model-View-ViewModel*. 2009-04-21 [Online; navštívené 2020-01-08].
URL https://www.dotnetportal.cz/clanek/4994/MVVM-Model-View-ViewModel

Príloha A

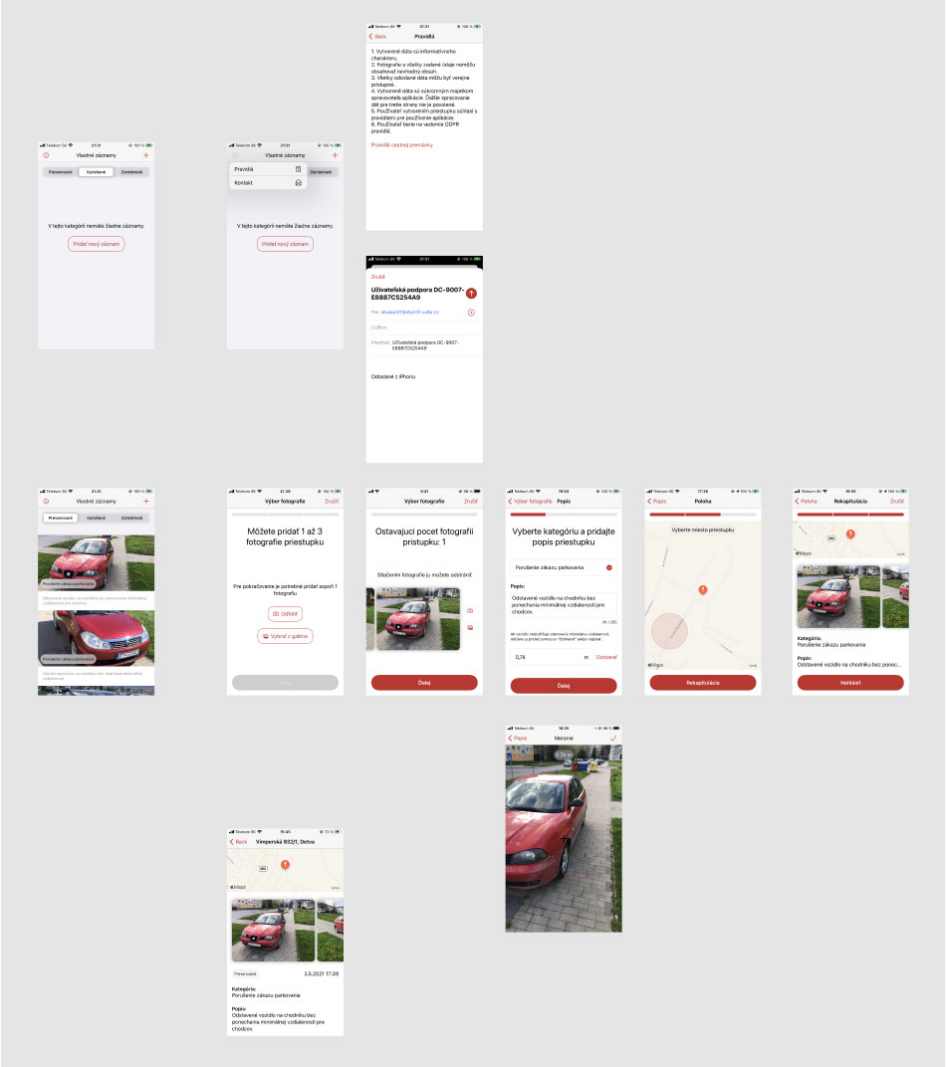
Plagát



Obr. A.1: Plagát pre výsledný systém

Príloha B

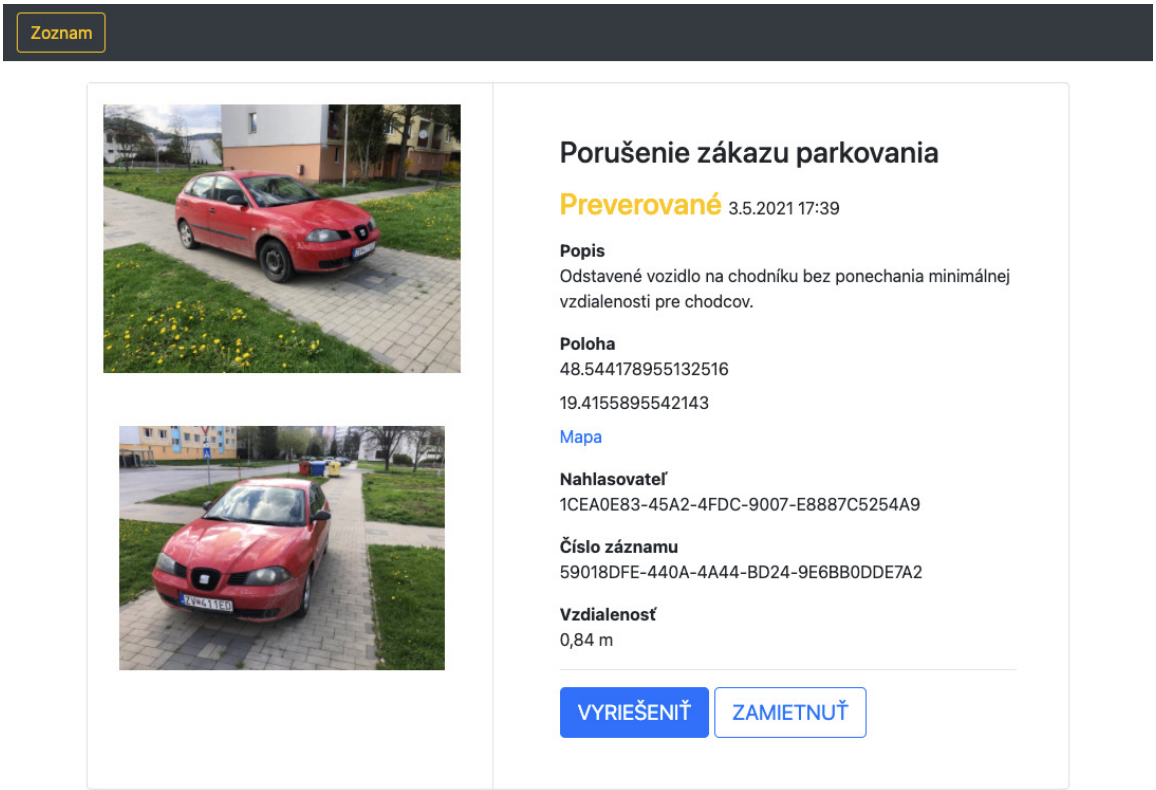
Prehľad obrazoviek mobilnej aplikácie



Obr. B.1: Všetky obrazovky mobilnej aplikácie na opačný systém iOS

Príloha C

Výsledný vzhľad webového rozhrania



Obr. C.1: Detail priestupku vo webovom rozhraní

Kategória	Popis	Poloha	Dátum vytvorenia	Stav
Porušenie zákazu parkovania	Odstavené vozidlo na chodníku bez ponechania minimálnej vzdialenosti pre chodcov.	48.544178955132516 19.4155895542143	3.5.2021 17:39	Preverované
Vozidlo stojace na zeleni	Vozidlo parkuje na zatrávnenej ploche pri Kapustách v Detve.	48.54966039918288 19.414965454427094	3.5.2021 17:47	Preverované

Obr. C.2: Zoznam priestupkov vo webovom rozhraní